Disambiguation Canvas: A Precise Selection Technique for Virtual Environments

Henrique G. Debarba[1,2], Jerônimo G. Grandi[1], Anderson Maciel[1], Luciana Nedel[1], and Ronan Boulic[2]

[1] Instituto de Informática, Universidade Federal do Rio Grande do Sul (UFRGS)
[2] École Polytechnique Fédérale de Lausanne (EPFL)
{jggrandi,amaciel,nedel}@inf.ufrgs.br,
{henrique.galvandebarba,ronan.boulic}@epfl.ch

Abstract. We present the disambiguation canvas, a technique developed for easy, accurate and fast selection of small objects and objects inside cluttered virtual environments. Disambiguation canvas rely on selection by progressive refinement, it uses a mobile device and consists of two steps. During the first, the user defines a subset of objects by means of the orientation sensors of the device and a volume casting pointing technique. The subsequent step consists of the disambiguation of the desired target among the previously defined subset of objects, and is accomplished using the mobile device touchscreen. By relying on the touchscreen for the last step, the user can disambiguate among hundreds of objects at once. User tests show that our technique performs faster than ray-casting for targets with approximately 0.53 degrees of angular size, and is also much more accurate for all the tested target sizes.

Keywords: Selection techniques, 3D interaction, usability evaluation, progressive refinement.

1 Introduction

Selection is one of the four fundamental forms of interaction in a virtual world [1, 2]. It is the ability of the user to specify objects in the virtual environment for subsequent actions [3]. The literature is rich in immediate selection techniques; however, this class of technique is exposed to problems of accuracy, ambiguity and complexity [4]. Many applications rely more on correctness of selection than on time of selection, but ordinary selection techniques in use tend to favor performance over accuracy. We intend to provide a precise yet fast alternative for selection in virtual environments. Therefore, we rely on selection by progressive refinement, as proposed by Kopper et al. [5].

Selection by progressive refinement proposes the breakdown of a selection task into "effortless" subtasks. It aims to avoid the attention and precision usually required by traditional selection techniques, so-called immediate selection techniques [5, 6]. However, there is an inevitable tradeoff between immediate and progressive refinement selection techniques. To complete a selection, the latter requires a process which

P. Kotzé et al. (Eds.): INTERACT 2013, Part III, LNCS 8119, pp. 388–405, 2013.
IFIP International Federation for Information Processing 2013

usually consists of more than one quick subtask, generally resulting in higher accuracy and longer selection time. On the other hand, immediate selection techniques consist in performing the selection in only one step, being generally faster but less accurate.

We propose the *disambiguation canvas*, which is a technique for quick disambiguation of selection. We use the observed high precision of control provided by the touchscreen [7, 8] to allow the disambiguation of the desired object among a subset of hundreds of other objects in only one step of refinement. Previous progressive refinement techniques do not scale as well as ours. Available techniques that disambiguate only in one step are limited to a small subset of objects, while those that refine among large subsets require multiple steps of disambiguation.

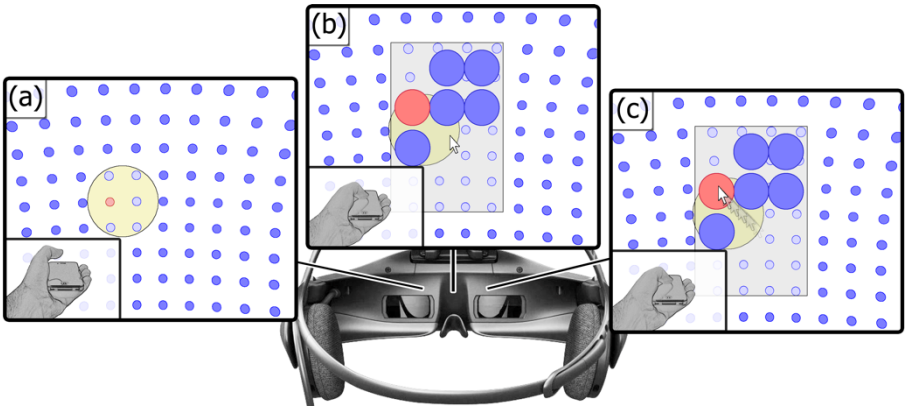

Fig. 1. Disambiguation canvas walkthrough: (a) the user points to the region where the desired object is located; (b) starting a touch rearrange the subset of objects pre-selected by the volume casting technique over a selection canvas; (c) the canvas has an absolute mapping to the mobile device touchscreen, the user slides his thumb in order to point out the desired object. Disambiguation canvas was designed to be compatible with immersive displays, such as the depicted head-mounted display. In this figure, the hand inserts illustrate the user gestures for each step, and are not displayed by our technique.

The *disambiguation canvas* is based on a two steps process. In the first step, the user employs a volume casting technique to point in the direction of the desired target object (see Figure 1a). When the target object is inside or intersecting the volume of selection, the user may start a touch on the mobile device touchscreen to enter the second step of the selection. A rectangle aligned parallel with the image plane – or with the mid orientation of the two image planes when stereoscopic rendering is in use – appears in front of the user; all the subset of objects pre-selected by the volume moves in an animation to form a matrix inside this rectangle (Figure 1b). The rectangle has a 1:1 mapping with the mobile device touchscreen, sliding the thumb on the touchscreen allows the superposition of the desired object by the arrow (Figure 1c). Selection is performed by a *take-off* gesture. If the user wants to leave the

disambiguation phase without selecting any object, they simply perform a *take-off* gesture with the arrow over an empty region of the rectangle.

The remainder of this paper is organized as follows. In order to provide the reader with a context, Section 2 summarizes the main selection techniques that consider progressive refinement; Section 3 presents the proposed technique design decisions, prototype hardware technology and software implementation. In Section 4 we present the evaluation of the *disambiguation canvas*, comparing it with the *ray-casting* and *SQUAD* [5], and present the analysis of results. A deeper discussion covering qualitative results as well as suggestions for design changes are presented in Section 5. Finally, in Section 6 we highlight our findings and suggest future developments.

2 Related Work

The design space of selection by *progressive refinement* was presented by Kopper et al. [5]. It is defined as an approach to progressively reduce the group of selectable objects and hence reduce required precision of pointing. From the current literature, we could identify three major groups of progressive refinement selection techniques: *menu disambiguation*, *zoom* and *persistence of pointing*. These are discussed below, as well as some techniques that do not fit these main categories.

Menu disambiguation: generally uses a volume of selection on the initial phase to reduce the effort of pointing into the desired object. Objects that fall inside or intersect that volume are then presented as a subset of objects using some sort of menu for disambiguation.

Dang et al. presented the *transparent sphere* and *transparent cylinder* [9]. On the transparent sphere technique, a positional cursor similar to the *virtual hand* metaphor [2] is used to place a spherical volume of selection in space. Objects inside or intersecting this sphere have their names shown on a menu. Disambiguation is performed by selecting the desired object name. On the other hand, the transparent cylinder uses a *ray-casting* based approach where a cylindrical volume is attached along the cast ray in order to define the subset of objects. Transparent sphere and cylinder present only the name of the target for disambiguation. Thus their original design is unsuitable for a series of applications.

Grossman and Balakrishnan [10] proposed the *flower ray* for interaction with a volumetric display. The *flower ray* uses *ray-casting*, and disambiguates using a marking menu. When entering the menu disambiguation step, intersected objects animate towards the user viewport and spread as a marking menu. However, this technique still requires precision of pointing as it relies on ray-casting for both phases, and would have problems disambiguating among a large subset of objects.

When proposing the taxonomy for the progressive refinement selection techniques, Kopper et al. [5] also presented the SQUAD technique (sphere-casting refined by QUAD-menu). SQUAD consists of defining a subset of objects through their intersection with a sphere volume, and further refining the subset through QUAD menus until only one object remains. As SQUAD relies on several steps of disambiguation, we believe the major drawback of this approach is that the visual search is repeated in

each step. If the desired object is similar to others, the visual search can be even more time consuming than the pointing task itself. This question was not addressed in the original study.

Zoom: Bacim et al. [6] propose two techniques for progressive refinement based on zoom, *discrete zoom* and *continuous zoom*. In the *discrete zoom*, the user defines a quadrant of the screen they want to see in more detail, the frustum changes so that the specific quadrant covers all the *field of view* (FOV). In the *continuous zoom* technique, the zoom happens continuously towards the pointing direction. Zoom based techniques have the advantage of showing the objects in their original context. However, manually controlled zoom tends to be more time consuming. Indeed, the evaluation presented by Bacim et al. showed lower performance when compared to the SQUAD technique.

Score accumulation: although not originally classified as progressive refinement selection techniques [5], we advocate that score accumulation techniques present the expected behavior described by the authors. These generally rely on the consistency of pointing, where objects that were targeted for a larger duration accumulate higher scores, becoming more likely to be the intended target of a selection.

Haan et al. [4] use an approach similar to *lightspot* (cone-casting) [11] for the *intenselect* technique. However, on *intenselect* an alternative disambiguation function is applied and expanded to the dimension of time. Objects that fall inside the cast cone accumulate scores over time. Their scores increase proportionally to their proximity to the center of the cone and its casting origin. If an object stops intersecting the cone, its score is gradually lowered. Visual feedback of pointing is given by a bended ray connecting the origin of the ray to the object with the higher score.

Grossman and Balakrishnan have implemented and evaluated the *smart ray* – technique first proposed by Steed [3] – on a volumetric display [10]. All objects intersected by the ray accumulate scores; to disambiguate, the user moves the origin and direction of the ray so that it always intersects the intended object. As long as the user succeeds maintaining the ray over the desired object for more time than any other, she is able to select it. *Smart ray* gradually decreases the score of the objects that have lost intersection with the ray. Therefore, it still maintains most of the score of objects that unintentionally lost contact with the ray for a short period of time.

Other approaches: Steed and Parker have proposed *shadow cone-casting* [12], which uses cone casting persistence of pointing along time to define a selection. When the user starts a selection, all objects inside the cone are selectable. The user must disambiguate among these objects by moving the origin of the cone while trying to always maintain the desired object(s) inside the cone. If an object falls outside the cone, it is cut out of this selection process. This technique allows the selection of multiple targets. However, it relies on the proximity of objects for this. Additionally, this technique is likely to be very time consuming in a cluttered environment, where high precision is required.

Grossman and Balakrishnan have proposed the *lock ray* [10], which expands their own technique of *depth ray*. The depth ray uses forward and backward hand movements to disambiguate which of the objects intersected by ray-casting will be selected. In the lock ray, these steps are performed in sequence, assuring higher precision

control as the ray-casting becomes locked, while in the depth ray they are performed simultaneously. Both techniques still require high precision of pointing in order to hit a target with the ray-casting metaphor.

Stellmach and Dachselt presented *Look & Touch*, which has more similarities with our approach [16], although being designed for 2D selection. With *Look & Touch* the user uses his gaze to control the direction of the lower precision phase of pointing; objects intersecting the circular area of their cursor are pre-selected, and the user can disambiguate cycling through the objects on an *iPod touch* touchscreen. Besides this disambiguation technique, they also propose a relative and an absolute control of the cursor using the touchscreen, similar to [17, 8].

3 Technique Design

This section exposes our design decisions while developing the *disambiguation canvas*, its particularities (such as objects distribution over the plane) and its implementation details.

3.1 Volume Casting Techniques

The most common approaches for volume casting are the *cone-casting* and the *sphere-casting* techniques. Which one of these is best fit for our technique may depend on the application. Thus, we decided to support both volume-casting techniques for the disambiguation canvas.

Using the cast of a sphere, it is likely that the amount of objects intersecting the volume is smaller, as the sphere has a limited depth. However, it is also harder to control the first step of the selection as the depth must be somehow provided; there are two common approaches to determine the sphere depth. The first uses the near intersection of the sphere, and sets its distance to the intersection position. The second casts a ray through the center of the sphere; the distance of the first intersection of this ray is used to set the sphere depth. In *SQUAD* for instance, Kopper et al. [5] favored the *sphere-casting* with the depth of the sphere determined by *ray-casting*, but this may be due to the type of environment for which they developed the technique, a virtual supermarket application. In the supermarket environment, objects were very cluttered and organized as stacks in many shelves. Such organization facilitates the task of pointing out a cluster of objects with *sphere-casting*.

On the other hand, *cone-casting* allows reaching objects even if an intersection occludes them from the casting origin point of view, which sphere-casting is unable to do. Cone-casting can also reduce the necessary precision during the first stage, as no depth input is required. However, *cone-casting* may intersect too many objects if the scene is very cluttered. In the supermarket case, *cone-casting* would require constraints in order not to select objects behind the shelves; otherwise a huge amount of objects may fall inside the conic volume.

Based on the report by [13] that users achieved up to 4° of error during the coarse precision phase of distal pointing, we suggest the angular size of 12° for the

sphere/cone casting technique. Six degrees from the center of the ray to the sphere/cone borders. The sphere/cone always rescales to achieve the angular size of 12° from the casting position point of view.

3.2 Graphic Representation

We use an *arrow* shape for the cursor, and a semi-transparent *rectangle* to represent the canvas. Based on its widespread use, we concluded that the arrow would be the most natural and intuitive representation for the cursor pointing position, while the rectangle helps to easily match and associate the mobile device touchscreen with an area on the virtual environment. These shapes are only visible during the second step, which is the disambiguation step. The mobile device pointing direction is represented by a semi-transparent sphere or a cone, depending on the volume-casting technique used. The volume-casting shape in use is always visible, so the user can always have feedback on his/her pointing direction, even when performing the second step of a selection (disambiguation step).

The rectangle uses an absolute mapping with the mobile device touchscreen. It is drawn to use 30° of the total 45° standard vertical FOV of the camera. We positioned it 70*cm* away from the camera on our immersive display implementation, thus we obtained the size of ≈38*cm*. However, in order to make better use of the FOV, this size should be decided according to the available display, allowing the user to inspect the objects more efficiently, and therefore reducing the visual search time. The distance on which it is drawn may also vary in order to avoid occlusion with other objects in the scene. In our specific implementation, we adopted the distance of 70*cm* to maintain a pleasant stereoscopic rendering when switching from background to canvas.

3.3 Mapping Objects to the Canvas

When entering the second step of the *disambiguation canvas*, the subset of objects must be reorganized side by side over the canvas plane. As most users might have trouble reaching the whole touchscreen with the thumb, we propose two standard layouts on which the objects are reorganized so the user can easily reach them. The first consists of ≈53.4% of the total area, and is oriented to user handedness (Figure 2a). This layout takes 5% from the right, the left and the top, and 25% from the bottom out of the useful area, as well as 1/8 and 1/16 of the remaining that is too close to the palm and far from the thumb reach respectively. The second layout consists of ≈42.4% of the total touch screen area. This layout takes 5% from the right and the left, 10% from the top, and 30% from the bottom out of the useful area. The final layout consists of a circle inside the remaining area, as illustrated in Figure 2b.

For a preliminary (not presented) and the first evaluation (Section 4.1), the layout presented in Figure 2a was used. Although in general it has worked properly for most subjects, two users from the preliminary evaluation had difficulty to reach a large portion of the layout, and in consequence obtained significantly higher error rates. Thus, we have also decided to approach a layout calibration method. To calibrate, the

user performs circular movements with the thumb on the touchscreen within their range of comfortable motion (Figure 3a). A flood fill algorithm identifies the outermost bounds for that user's specific layout. This approach was used in the second evaluation, presented in Section 4.2.

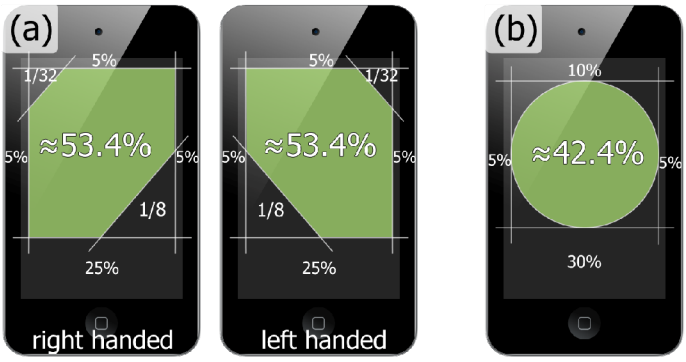

Fig. 2. Standard layouts of useful touchscreen area proposed for the disambiguation canvas techniques. The (a) standard layout was used for the first presented evaluation.

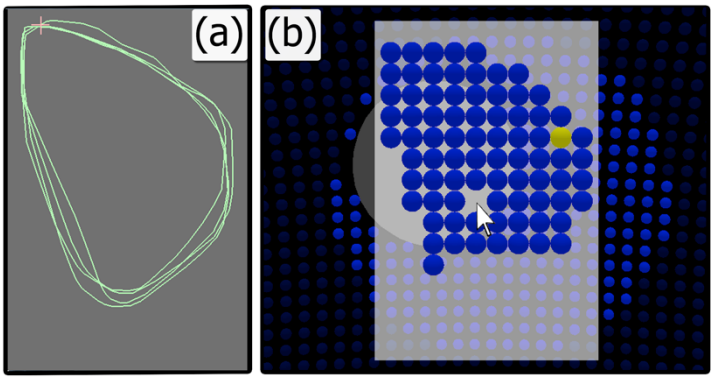

Fig. 3. Proposed layout calibration process, (a) contour of the reachable area defined by a user. (b) Arrangement of objects inside the reachable area, notice that the starting position of the arrow is kept empty. This procedure was used for the second evaluation (Section 4.2).

When switching to the disambiguation phase, a matrix fitting every pre-selected object inside the layout is computed, and each object is designated to a valid slot of the matrix (Figure 3b). A slot is considered valid if its center is located inside the usable area defined by the layout. In order to fit every object inside their designed slot, the objects are rescaled so their bounding box does not pass beyond that space. However, this may make some visual attributes less apparent or even impossible to be perceived, such as when the user wants to select a target of specific size within a group of similar objects. To overcome this issue the rescale factor may also be proportional to the largest and smallest target, being linearly remapped between a minimum and maximum threshold. That is, if the smallest object is ten times smaller than

the largest one, this proportion factor (1:10) will be lowered to a maximum of half of the size (1:2) so both objects remain visible and distinguishable. Objects with intermediate sizes are proportionally rescaled in between these thresholds. By default, our current implementation uses this approach.

In our current implementation, which was used for the second user evaluation (Section 4.2), the position where the user starts a touch is not superposed by objects. This design facilitates leaving a selection procedure if the user does not want to select any of the objects pre-selected by the first phase (Figure 3b). In addition, it also avoids accidental selection in the case of an unintentional touch by the user.

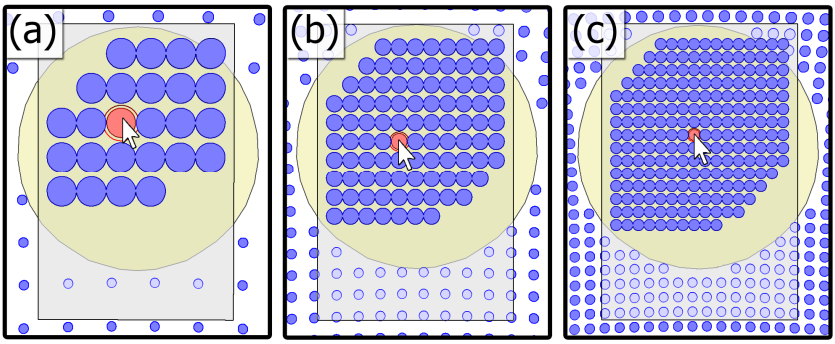

Fig. 4. Difficulty of selection is proportional to the amount of objects pre-selected by the first step; in (a) there are 25 objects on the canvas, in (b) 97 objects, and in (c) 224 objects.

A regular immediate selection technique usually has its difficulty of pointing increased by reducing the target object size. However, when using the *disambiguation canvas* the difficulty increases according to how many objects have been pre-selected by the volume-casting during the first phase. This is an expected behavior of progressive refinement selection techniques, and might make the refinement slower and/or harder. However, as the *disambiguation canvas* relies on the mobile device touchscreen for disambiguation, we are able to align hundreds of objects within a single disambiguation step while still ensuring high precision. During the first technique evaluation (Section 4.1), we have used three distinct object densities; they are shown in Figure 4. For the worst case depicted in Figure 4c, 224 objects went to the disambiguation phase. Still, our technique offered the very convenient sensing area of ≈230 pixels of the touchscreen (the orthogonal projection of the spherical object over an area of 19×19 pixels), which has a total input area of 320×480.

3.4 Prototype Implementation

We used an *Intel Core i7* computer, equipped with two *AMD Radeon HD 5870 Eyefinity6*. The immersive display is a *Sensics zSight Integrated SXGA Head Mounted Display* (HMD) (Figure 5a). It provides stereoscopic vision using two 1280 x 1024 displays, and has a FOV of 60°. This HMD also provides the orientation of the head. For the mobile device, we have chosen the *Apple iPhone 4/4S* and *iPod touch 4*.

The software displayed by the *zSight HMD* is implemented in *C++*, using *Ogre3D* for graphics [14] (Figure 5b). We support stereoscopy in our application. The mobile device software is an *app* implemented in *Objective-C*. It acquires the sensor readings and communicates them over a Wi-Fi infrastructure through *UDP*. To obtain the orientation of the *iPhone 4/4S*, which has a *magnetometer* – thus providing the recalibration of drift on the *yaw* – we have used the strategy proposed by Madgwick [15] with the adaptations presented in a previous work [8]. In [8], a complete description on the acquisition and processing of sensor data to provide orientation is presented. To obtain the orientation of the *iPod touch 4*, which does not contain a *magnetometer*, we have used the standard orientation provided by the *iOS SDK*.

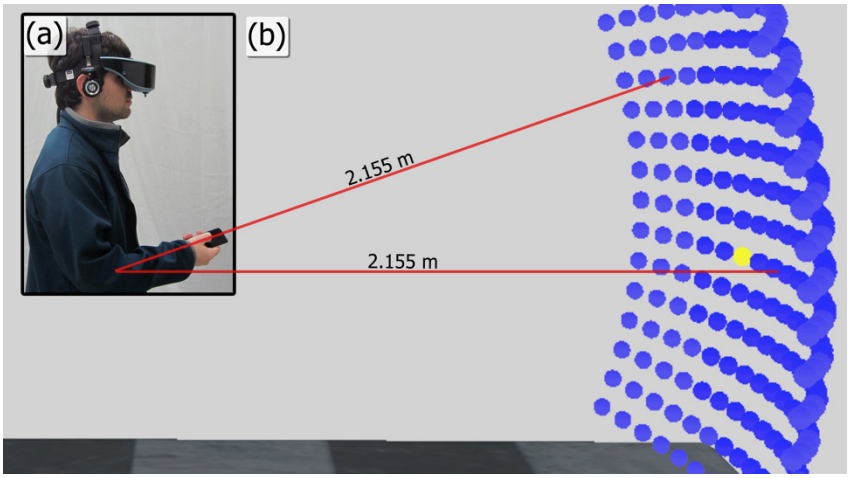

Fig. 5. Merged images of the prototype overview (a), and the test application used for evaluation (b)

4 Disambiguation Canvas Evaluation

We conducted two sets of user tests for the *disambiguation canvas* technique. The main design decisions are common to both evaluations and were based on those used by Kopper et al. for the SQUAD technique evaluation [5]. Both were comparative evaluations and used a within-subject design. In the first we compare *disambiguation canvas* with *ray-casting*, while in the second we compare it with *SQUAD*. The implemented ray-casting relies only on the orientation of the device – assumption of a constant casting position – and is therefore referred to as *ORayCasting* (orientation ray-casting); *disambiguation canvas* and SQUAD also use only the orientation of the device for the volume-casting step. The *disambiguation canvas* is referred to as *DCanvas*. For both evaluations the objects pre-selected by the first phase of DCanvas were animated from their original position to the canvas in a 250ms animation.

An *iPod touch 4* was used in both evaluations. As mentioned earlier, it is not equipped with a magnetometer, and thus is subject to drift on *yaw*, losing its correct

orientation. Therefore we used blocks with no more than 11 trials, and applied an offset in order to correct any accumulated error between blocks. On the other hand, it produces an orientation less noisy than using the magnetometer to correct yaw, necessary for reliability of pointing while using ray-casting. Mobile device orientation was filtered using a *dynamic low-pass filter*, interpolating between cutoffs of 0.2Hz and 50Hz, with 60Hz sampling rate. Cutoff is defined according to the angular change speed in degrees: when < 1°/*sec*, the lowest cutoff is used (0.2Hz); when > 50°/*sec*, the highest cutoff is used (50Hz). For any speed between those, a linearly interpolated cutoff value is used.

General goal: the goal was to select a yellow sphere among several distractors of same size represented as blue spheres. These objects were arranged as a matrix. To position them, we use a main sphere of 2.155*m* radius. All the selectable spheres were positioned with their centers intersecting the borders of this main sphere. The origin of the ray/sphere-casting was set to the center of the main sphere. This guarantees the same angular pointing size for all the objects. The virtual camera was positioned 50*cm* above the ray/sphere-casting origin. This configuration is shown in Figure 5.

Evaluation procedure: The same procedure and similar questionnaires were used for both of our evaluations. The procedure was as follows:

1. The subject was asked for any health issue or impairment that could prevent them from participating (such as a history of epilepsy and color blindness)
2. The subject filled in a characterization questionnaire
3. The subject was presented to the first technique on an ordinary screen display (so the experimenter and the subject could share the view while explaining how the technique works)
4. The experimenter presented the HMD and how to adjust it to the head
5. The subject performed practice blocks with the first technique
6. The subject performed evaluation blocks with the first technique
7. The subject answered a questionnaire about the first technique
8. The subject repeated steps 3, 5, 6 and 7 for the second technique
9. The subject filled in a post-experiment questionnaire comparing both techniques

A block consisted of a collection of trials. A trial consisted of a selection task, ending with an activation of selection, which could be successful or not. The number of blocks and trials is different for each evaluation. Subjects were allowed to remove the HMD and rest between the blocks if desired.

4.1 Comparison with ORayCasting

Design: sphere-casting was used for the volume-casting step of selection on DCanvas. Instead of using the suggested standard size of ≈12° for the casted sphere, we used the angular size of ≈26°, so more objects would be pre-selected for the subsequent disambiguation phase. This sphere is represented by a semitransparent sphere. The ORayCasting casted ray is represented by a cylinder with 1*cm* of diameter.

The independent variables are *angular space* between objects (resulting in different distractors density): 5°, 2.5° and ≈1.67°; and *angular size*: ≈0.53°, ≈1.06° and ≈1.6° (2*cm*, 4*cm* and 6*cm* respectively). We used blocks of 10 trials, 9 of them representative of the combination of *density* × *size*, and an initial target which was used to start the block. Training consisted of 5 blocks, while the evaluation consisted of 10. The 9 valid targets within each block were randomly presented, while technique presentation order was counterbalanced.

The target was randomly chosen among the objects with only one constraint, this object should have its center within a range between 52*cm* and 77*cm* from the center of the matrix of objects. We did so in order to keep the possible targets within the user's field of view, thus reducing visual search bias. We have also colored objects farther than 77*cm* to green, so the user knows they are not target candidates. For this evaluation, the collision checking on the disambiguation step was performed using the superposition of the arrow over the projection of the sphere on the canvas (a circular area).

The comparison with *ORayCasting* was also intended to verify design decisions and to evaluate whether the technique was comprehensive and easy to use. The design choices presented in Figure 3 were not used in this evaluation. In fact, they were implemented after the feedback from this experiment and were used for the following *SQUAD* comparison (Section 4.2).

Subjects: six graduate students in Computer Science from our university participated in this experiment (mean age of 29, four right handed). All of them were very experienced in managing mobile device touchscreens, and had at least some experience using natural pointing devices. In a 7 points scale, only two reported experience with virtual reality equipment of 3 or more points. Each test took from 25 to 40 minutes to be performed. We have obtained a total of 1,080 valid trials: *2 techniques* × *6 subjects* × *10 blocks* × *9 trials*.

Results: overall mean selection times with *DCanvas* and ORayCasting were respectively: 2.37 and 2.29 seconds. One-way ANOVA showed that *DCanvas* was slower with statistical significance when compared to *ORayCasting* ($F(1.1078)=4.43$, $p<0.036$). See Figure 6 for detailed mean time for each combination of target size and density. Error rate with ORayCasting was significantly higher than with *DCanvas* ($F(1.1078)=70.34$, $p<0.0001$). Error rates for each combination of size and density are presented on Figure 6.

Figure 7 presents the time and error rate per user. For this experiment, *DCanvas* obtained a lower mean time for two users. Four users have not made any selection error while using our technique. Subject 6 presented an exceptionally low error rate for *ORayCasting*, but still higher than with *DCanvas*.

The intermediate questionnaire asked users to rate each technique concerning: ease of learning and ease of use; how well it performs for small, medium and large targets; and how much fatigue was felt on their wrist, hand, fingers, back and legs. Results are presented in Figure 7. Both were considered very easy to learn, while our technique was considered easier to use. *DCanvas* was preferred over *ORayCasting* for small and

medium targets, while large targets received equivalent ratings for both techniques. Overall fatigue was lower for *DCanvas*; its mean of the 5 related questions was 2.1, against 2.5 of *ORayCasting*.

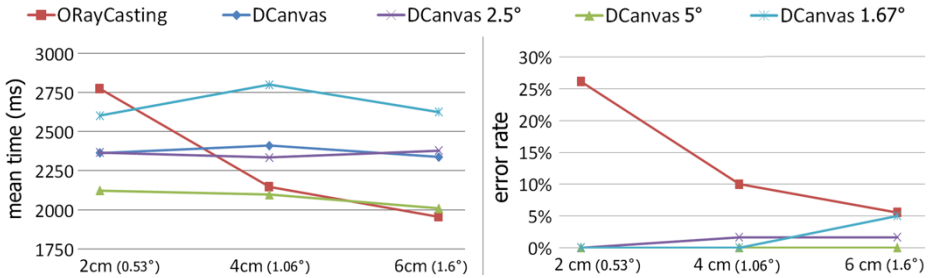

Fig. 6. Mean trial completion time and error rate for each combination of target angular size and density for the comparison with ORayCasting

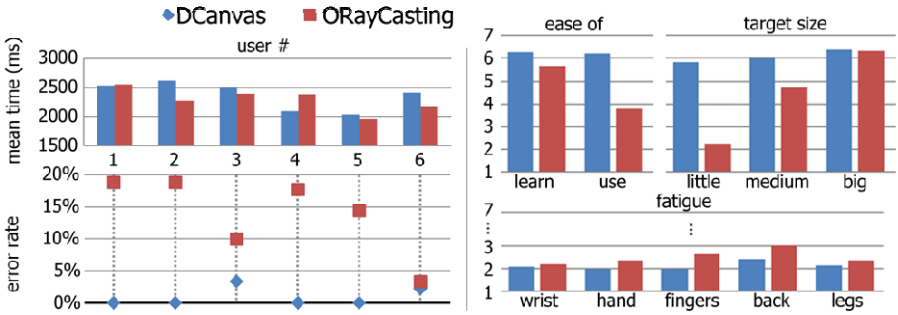

Fig. 7. Mean time and error rate per user and subjective questionnaire scores for DCanvas and ORayCasting

Regarding the comparative questionnaire, *DCanvas* presented higher scores for all the questions. It was considered more accurate (6.7 against 1.6), faster (5.9 against 2.7), less tiring (4.8 against 2.6) and easier to use (5.6 against 2.5). Curiously, users felt that *DCanvas* was faster, which is true for ≈.53° angular size targets, but false for the overall evaluation. *DCanvas* was also preferred by all the users.

4.2 Comparison with SQUAD

Design: for the comparison between DCanvas and SQUAD we have eliminated the target size from the set of independent variables. Our previous evaluation, as well as the one performed by Kopper et al. to test SQUAD, showed that the target size is not significant for time of selection or error rate. Thus we used the constant angular size of ≈1.06° (4*cm*) for all the objects. For the independent variable of angular space between objects (distractors density) we used: 6°, 3°, 2°, 1.5° and 1.2°.

We used blocks of 11 trials, 2 trials of each density, which were randomly presented, and an additional initial target used to mark the start of the block. Training

consisted of 3 blocks, while the evaluation consisted of 4. Technique presentation order was counterbalanced. The target object was randomly chosen with the same constraint as before, but accepting a range between 45*cm* and 55*cm* from the center of the matrix of objects. Objects beyond 60*cm* from the center were colored in dark blue, which is less distractive than the green used for the previous evaluation.

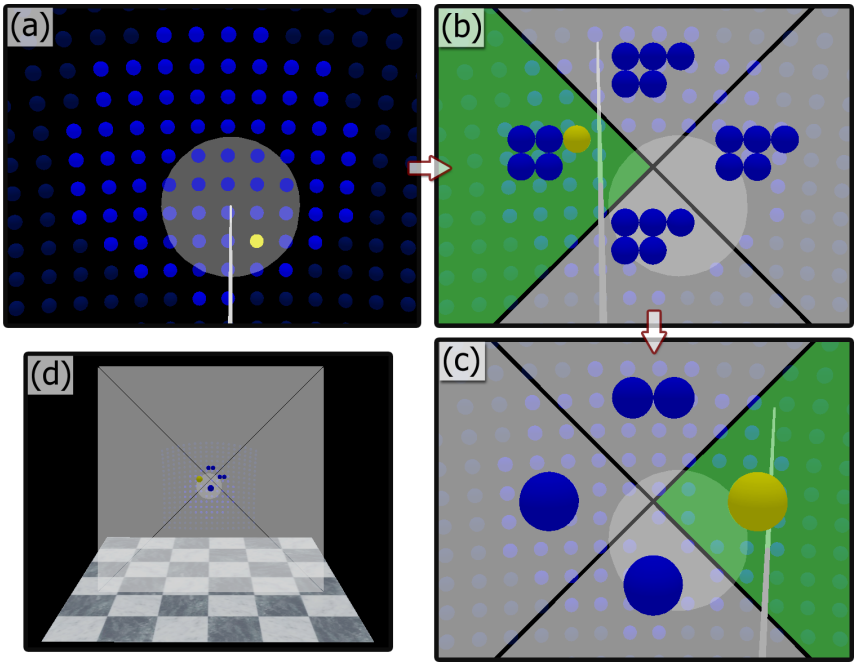

Fig. 8. SQUAD implementation used for comparison: (a) the group of objects inside the sphere-casting volume is selected by a tap gesture; (b) these objects are rearranged into quadrants in a 250ms animation, to select a quadrant the user perform a tap gesture while intersecting it with ray-casting; (c) the subgroup is rearranged in new quadrants with a 200ms animation, a new tap gesture while intersecting the target quadrant results in a successful selection. Note that although the animations impose some time constraints, it also avoids the need for visual search at each new step, and the user can start the repositioning of the ray during the animation. (d) Overview of the quadrants area of selection, a tap gesture while not intersecting any quadrant – or intersecting an empty quadrant – may be performed to leave the selection procedure.

We used *sphere-casting* for the first step of selection with DCanvas and SQUAD. Instead of adopting the suggested standard size of ≈12° for the casted sphere, we used the angular size of ≈17° for both techniques, so more objects would be pre-selected for the subsequent disambiguation phase. Combined with the possible angular space between objects for this evaluation, the sphere-casting phase could pre-select ≈6, ≈25, ≈55, ≈100 or ≈160 objects for the disambiguation phase. The sphere-casting step of the DCanvas and SQUAD was represented by a semitransparent sphere.

The SQUAD casted ray – disambiguation phase – was represented by a cylinder with 1*cm* of diameter. Figure 8a-c presents the walkthrough of our SQUAD

implementation. The quad menu is drawn at a distance of 100*cm* from the camera, and is oriented to face the camera. The quad menu is composed of four triangles that assemble a square of 4×4 meters (Figure 8d). To recover from a mistaken pre-selection, the user can point outside of the quad menu and perform a tap gesture, or select an empty quadrant. An error only occurs when the user selects a wrong quadrant that contains only one object inside.

For this evaluation, the collision checking on the disambiguation step of DCanvas was performed using the superposition of the arrow over the designated objects slot (instead of the object projection) in the canvas. This approach increases the effective selection size of the object, and also allows a simpler collision test when dealing with objects with a mesh more complex than those tested (spheres). Additionally, the improvements presented in Figure 3 were also used.

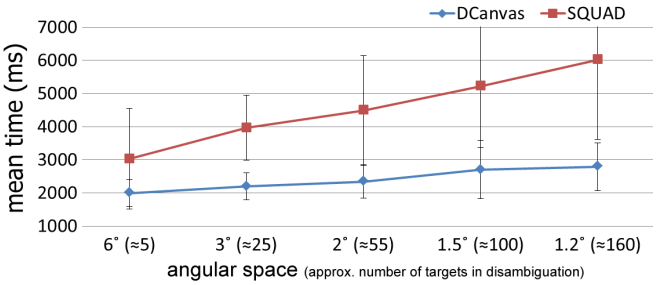

Fig. 9. Mean trial completion time for each angular space (density) of objects distribution for the comparison with SQUAD

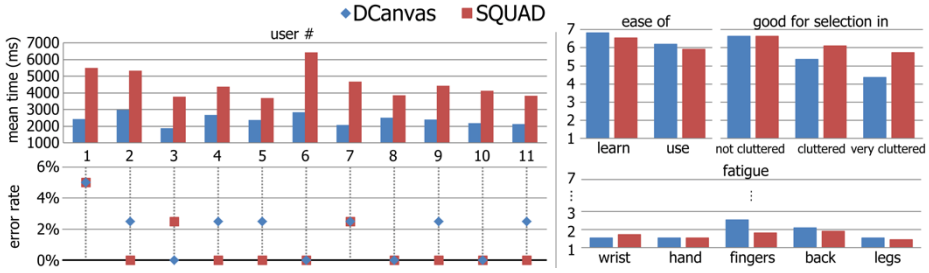

Fig. 10. Mean time and error rate per user and subjective questionnaire scores for the comparison with SQUAD

Subjects: eleven subjects participated on this evaluation, all of them students or professors in computer science or electrical engineering (mean age of 31). On a seven points scale, only one reported little experience with mobile device touchscreens (below 5), and three reported experience with pointing devices equal or above 4 points. Only one subject reported high frequency of use of virtual reality devices.

Results: as demonstrated by Figure 9, DCanvas performed significantly faster than SQUAD for all conditions. We also highlight that the increase in selection time for DCanvas was less steep than SQUAD. For the error rate, both techniques achieved

good marks. DCanvas had an error rate of 0.018 errors per trial, while SQUAD achieved 0.009 errors per trial. Figure 10 show the individual performance of mean time and error rate for each user.

Concerning the comparative questionnaire, 6 subjects preferred the DCanvas, while 5 liked SQUAD most. The mean scores of comparative questions were very similar between techniques. DCanvas was considered less precise by a difference of ≈1.26 points on the 7 points Likert scale, and more difficult to learn and use by a difference of ≈0.45 and ≈0.39. DCanvas was regarded as faster by a difference of ≈0.19.

However, on the questionnaire specific for each technique, when no direct comparison was required, the DCanvas obtained higher scores concerning its ease of learn and use (Figure 10). On the other hand, SQUAD received higher absolute scores concerning the level of cluttering of the environment. Concerning the fatigue, answers were generally very similar, except by the fatigue on the fingers, which was higher for DCanvas. The mean of these scores are reported in Figure 10.

5 Discussion and Final Remarks

5.1 Transition to the Canvas

The most recurrent feedback left by the users regards the transition of the subset of objects from its original context to the control canvas. Users frequently had the conviction that positioning the intended target near the center of the sphere during the sphere-casting step would take that target near the center of the canvas when switching to disambiguation. This intuition may arise from the arrangement of objects as a matrix, which would be easily fitted inside the layout. However, on a more complex scenario, with targets spread in depth, such organization is not so obvious. We are currently working on this issue, as it could reduce user effort of reaching objects mapped to distant regions of the touchscreen and reduce visual search time.

5.2 Keeping the Context

As for being a progressive refinement technique based in menu disambiguation, objects that go from the first to the second phase lose their original context. This could make it difficult to distinguish the intended object in real applications if they are very similar in shape or if the selection depends on their original topology. We propose three possible solutions for such limitation of the menu disambiguation approach. The first solution is to control the instant of interpolation that animates the objects while bringing them over to the disambiguation menu. To cast a ray or a volume for pointing, only 2 degrees of freedom (DOF) among the 3 provided by the device orientation are required. Our proposal is to use the 3rd DOF to dynamically control the instant of the interpolation. The mapping from orientation into instant of interpolation can be achieved with an absolute relation, where a certain orientation always results on the same instant, or with a relative relation, where after a threshold the orientation controls acceleration forward or backward on the interpolation instant. This strategy was

implemented, and showed to be functional; however it was not yet evaluated. Figure 11 shows four frames of an animation, moving the objects from their original to new position in the disambiguation canvas.

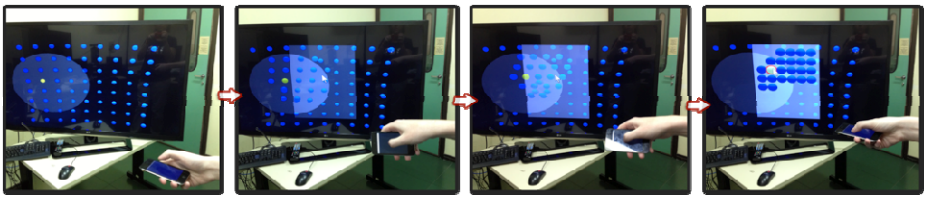

Fig. 11. Four frames illustrating the animation of the objects from their original positions to the new ones on the disambiguation canvas. The animation can be reproduced in both directions by twisting the mobile device in the corresponding direction.

The second technique consists in duplicating the original object into the canvas – instead of moving the original – and using the copy superposed by the arrow cursor to highlight the original object. It can indicate whether the user is pointing to the desired object when there is more than one object with the same or similar shapes.

The third approach is to draw a trajectory curve to connect the original position of an object with its final position on the canvas. This approach allows the simultaneous observation of all connections between original and final positions at the same time. However, this can result in cluttering and may overwhelm the user with information when too many objects are taken for disambiguation.

Notice that these suggestions are not exclusive and can be combined among them. Given that this is a general problem of menu based progressive refinement selection techniques, we intend to investigate these approaches further in future works.

5.3 Immersive Tool

The tests we led, as well as the prototype implementation described in Section 3.4 considered the use of a head-mounted display (HMD) and 3D stereo visualization to enhance the realism and immersion. Subjects were comfortable with this set up. However, even if the camera is driven by the movement of the user's head, we are not taking much advantage of this, since the objects were concentrated in a relatively small area, in front of the user. Our objective so far was to compare the disambiguation canvas with other techniques – as ray-casting and SQUAD – accordingly. Then, we tried to avoid any other independent variable.

Informally, we also tested the disambiguation canvas with a regular display. Our intuition is that its use with a regular display depends strongly on the layout of the objects. On the other hand, using the HMD provides easy control of the camera, which comfortably overcomes this limitation.

We are also aware that, to verify the robustness of the technique, more tests should be done with other layouts for the objects in the scene. Currently, they are all disposed on the surface of the sphere that surrounds the user.

6 Conclusions and Future Work

In this paper we have presented the *disambiguation canvas*, a technique for fast and high accuracy selection of objects in a 3D space. It relies on *progressive refinement* to address the lack of accuracy common to immediate selection techniques, and uses distinct input hardware to optimize the control over its two steps and overall time performance on a selection procedure.

By using the touchscreen on the second step of the *disambiguation canvas*, users were able to select objects represented in a motor area of $\approx 7.9mm^2$ of the touchscreen surface very efficiently during evaluation, allowing consistent disambiguation among a group of 150≈250 objects. However, the limits for efficient disambiguation with our technique are still unknown. Perhaps the simultaneous exhibition of so many objects to the user may be more limiting than the precision of input of the mobile device touchscreen. In [8], precision above 60% was obtained on a touchscreen area as small as $\approx 0.6mm^2$. If we transfer this parameter to the *disambiguation canvas*, the whole device touchscreen surface would allow the disambiguation of up to 6,144 objects in one step. Are we able to display meaningful objects in the order of thousands to the user? Is the user able to search for a specific object within such a large group? If this is the case, we also intend to adapt our technique to ensure a reliable selection with one additional step, such as an area selector that points a group of objects within a radius from the thumb position. With this strategy, and taking advantage of the touchscreen precision, we expect to reduce the selectable objects by a factor of at least 10, instead of the factor of 4 used by SQUAD.

Nevertheless, we emphasize that the user's subjective rating assumes that our technique was faster than *ray-casting*, while it had in fact performed slightly slower. This might be a clue to how unpleasant it is to perform a difficult selection with full attention. We have observed that even the breath had to be controlled for some users.

Acknowledgments. We would like to thank CNPq-Brazil for the financial support through projects 311547/2011-7, 485820/2012-9, 302679/2009-0 and 305071/2012-2. Thanks are also due to Microsoft Interop Lab at UFRGS. We finally thank the volunteers who kindly agreed to participate in the experiments.

References

1. Bowman, D.A., Kruijff, E., LaViola, J.J., Poupyrev, I.: 3D User Interfaces: Theory and Practice. Addison Wesley Longman Publishing Co., Inc., USA (2004)
2. Mine, M.: Virtual environment interaction techniques. Technical report, UNC Chapel Hill CS Dept. (1995)
3. Steed, A.: Towards a general model for selection in virtual environments. In: Proceedings of the 3D User Interfaces, 3DUI 2006, USA, pp. 103–110. IEEE Computer Society (2006)
4. Haan, G.D., Koutek, M., Post, F.H.: Intenselect: Using dynamic object rating for assisting 3d object selection. In: Virtual Environments 2005, pp. 201–209 (2005)

5. Kopper, R., Bacim, F., Bowman, D.A.: Rapid and accurate 3d selection by progressive refinement. In: Proceedings of the 2011 IEEE Symposium on 3D User Interfaces, 3DUI 2011, pp. 67–74. IEEE Computer Society, USA (2011)
6. Bacim, F., Kopper, R., Bowman, D.: Design and evaluation of 3d selection techniques based on progressive refinement. International Journal of Human-Computer Studies (to appear, 2013)
7. Baudisch, P., Chu, G.: Back-of-device interaction allows creating very small touch devices. In: Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, USA, pp. 1923–1932. ACM (2009)
8. Debarba, H., Nedel, L., Maciel, A.: Lop-cursor: Fast and precise interaction with tiled displays using one hand and levels of precision. In: 2012 IEEE Symposium on 3D User Interfaces, 3DUI, pp. 125–132 (2012)
9. Dang, N.T., Le, H.H., Tavanti, M.: Visualization and interaction on flight trajectory in a 3d stereoscopic environment. In: The 22nd Digital Avionics Systems Conference, DASC 2003, vol. 2, pp. 9.A.5 –91–10 (2003)
10. Grossman, T., Balakrishnan, R.: The design and evaluation of selection techniques for 3d volumetric displays. In: Proceedings of the 19th Annual ACM Symposium on User Interface Software and Technology, UIST 2006, pp. 3–12. ACM, USA (2006)
11. Liang, J., Green, M.: Jdcad: A highly interactive 3D modeling system. Computers & Graphics 18(4), 499–506 (1994)
12. Steed, A., Parker, C.: 3D selection strategies for head tracked and non-head tracked operation of spatially immersive displays. In: 8th International Immersive Projection Technology Workshop (2004)
13. Argelaguet, F., Andujar, C.: Visual feedback techniques for virtual pointing on stereoscopic displays. In: Proceedings of the 16th ACM Symposium on Virtual Reality Software and Technology, VRST 2009, USA, pp. 163–170. ACM (2009)
14. Ogre 3D: Ogre 3D (2012), `http://www.ogre3d.org`
15. Madgwick, S.O.H., Harrison, A.J.L., Vaidyanathan, R.: Estimation of IMU and MARG orientation using a gradient descent algorithm. In: 2011 IEEE International Conference on Rehabilitation Robotics (ICORR), pp. 1–7 (2011)
16. Stellmach, S., Dachselt, R.: Look & touch: gaze-supported target acquisition. In: Proceedings of the 2012 ACM Annual Conference on Human Factors in Computing Systems (CHI 2012), pp. 2981–2990. ACM, USA (2012)
17. Nancel, M., Wagner, J., Pietriga, E., Chapuis, O., Mackay, A.: Mid-air pan-and-zoom on wall-sized displays. In: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, CHI 2011, pp. 177–186. ACM, USA (2011)