HAL
archives-ouvertes.fr

Semantic Pointing for Object Picking in Complex 3D Environments

Niklas Elmqvist, Jean-Daniel Fekete

▶ **To cite this version:**

Niklas Elmqvist, Jean-Daniel Fekete. Semantic Pointing for Object Picking in Complex 3D Environments. Proceedings of Graphics Interface 2008, May 2008, Windsor, ON, Canada. ACM, pp.243-250, 2008, <http://dl.acm.org/citation.cfm?id=1375755>. <hal-00851711>

HAL Id: hal-00851711
https://hal.inria.fr/hal-00851711

Submitted on 16 Aug 2013

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Semantic Pointing for Object Picking in Complex 3D Environments

Niklas Elmqvist* Jean-Daniel Fekete†

INRIA

ABSTRACT

Today's large and high-resolution displays coupled with powerful graphics hardware offer the potential for highly realistic 3D virtual environments, but also cause increased target acquisition difficulty for users interacting with these environments. We present an adaptation of semantic pointing to object picking in 3D environments. Essentially, semantic picking shrinks empty space and expands potential targets on the screen by dynamically adjusting the ratio between movement in visual space and motor space for relative input devices such as the mouse. Our implementation operates in the image-space using a hierarchical representation of the standard stencil buffer to allow for real-time calculation of the closest targets for all positions on the screen. An informal user study indicates that subjects perform more accurate pointing with semantic 3D pointing than without.

Keywords: 3D picking, motor space, visual space, target acquisition, Fitts' law

Index Terms: H.5.2 [Information Interfaces and Presentation]: User Interfaces—Interaction styles; I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism

1 INTRODUCTION

Modern advances in display and graphics hardware have certainly raised the bar for realistic and highly dynamic 3D environments for computer games and interactive simulations. However, a less recognized fact is that this increased display resolution and detail level also cause problems for the users interacting with the environments. As potentially graspable game objects shrink in size in relation to the whole display, the task of acquiring and picking the objects becomes more difficult (see Figure 1 for an example from the 3D real-time strategy game *Spring*).

Semantic pointing [7] has previously been suggested as an approach to improve target acquisition for 2D user interface elements by modifying the pointer gain depending on its proximity to the clickable elements. In this paper, we adapt the basic concept of semantic pointing to a 3D context where the potential targets change as the user moves through the 3D environment and objects enter and leave the current view. The idea is to simply utilize knowledge of potential targets that is already available but typically ignored by current 3D applications. Furthermore, we present an approach to use the actual bitmapped projections of targets onto the image space as opposed to the regular geometric figures from earlier work on semantic pointing.

This paper presents the following main contributions: (a) an adaptation of semantic pointing to 3D object picking; (b) a real-time image-based implementation of semantic 3D pointing for dynamic 3D environments; and (c) results from an informal user study comparing the performance of object picking in two different game-like 3D environments with and without semantic 3D pointing.

Figure 1: Example of complex 3D environments from the Open Source real-time strategy game *Spring*. Note the high number of small units that the user has to select in order to control.

We continue this paper with a treatment of the related work. We then describe semantic 3D pointing and present a case study on employing it in a 3D game. We continue by outlining our informal user study and the results we collected from it. We end the paper with a discussion, our conclusions, and some words on future work.

2 RELATED WORK

Target acquisition is an important problem in human-computer interaction and has been studied by many researchers. However, most existing techniques are based on 2D pointing, such as in windowing systems. In the following section, we will describe the state of the art starting with 2D interaction techniques and then moving on to the 3D case.

2.1 Pointing in 2D

Fitts' law [12] is widely used for modeling human performance for target acquisition in human-computer interaction [18]. It gives an indication of task difficulty as a function of target size and its distance from the pointer. The most straightforward way to support target acquisition is then to either expand the size of the target, or to reduce the distance. Balakrishnan [3] offers an excellent overview

*e-mail: elm@lri.fr

†e-mail: jean-daniel.fekete@inria.fr

of a wide array of approaches for improving target acquisition; the most relevant are presented below.

Increasing target size in visual space causes the neighborhood to be shrunk or distorted and is therefore only possible in specialized circumstances. Furthermore, it may not be at all possible on a mobile device with a small screen. McGuffin and Balakrishnan [20] investigated a mechanism for dynamically expanding targets from the viewpoint of Fitts' law and showed that target acquisition performance is improved even if the expansion happens only in the last 10% of the pointer movement.

Fisheye views [13] have been employed to locally expand visual space around a focus (typically the pointer) and thus improve target acquisition performance in that area. Fisheye menus [6] and the MacOS X fisheye dock are examples of this. However, the fisheye view itself imposes a non-linear transform on the visual display, which means that the display is not stable under motion of the focus [16]. Furthermore, the visual distortion does not carry over to motor space, which means that a small movement in motor space can have an unexpectedly large impact in visual space and prompt "hunting" effects. Taken together, these effects can be disorienting and detrimental to target acquisition performance [28].

Coupling visual and motor space has been shown to yield better performance than mere visual expansion [20, 28]. However, Cockburn and Brock [10] showed that visual expansion offers reliable performance improvements over simple highlighting, and argue that some of the performance gains in previous techniques may be attributed to this fact and not to the increase of target size in motor space.

While most of the above techniques focus on increasing target size, the drag-and-pop [4] technique instead decreases the distance. This is done by moving potential targets that are in the trajectory of the cursor towards it. Object pointing [15] takes this concept to its extreme, providing a void-phobic extra cursor that jumps between potential targets that are selected based on the mouse trajectory.

Finally, some approaches perform the distortion only in motor space, invisibly reducing distances and expanding targets. The technique presented in this paper is directly inspired by semantic pointing for 2D [7], but other techniques to create the same effect include force fields for improving selection in pull-down menus [1], snapping support for object alignment [5], sticky and bubble targets [11], control-display gain optimization [19], and area cursors and sticky icons for supporting aging users [27].

2.2 Pointing in 3D

Simple visual target expansion in 3D using scaling causes objects to grow larger with no regard to their surroundings, and is thus unsuitable for the kind of realistic 3D scenes discussed in this paper. Regardless, some of the above techniques can be applied directly to 3D object picking; for example, Steinicke and Hinrichs [24] describe an adaptation of drag-and-drop to 3D called grab-and-throw that uses snapping to lock thrown objects in position.

In general, 3D object interaction techniques are divided into *egocentric*, *exocentric*, and *hybrid* techniques depending on their frame of reference [2]. While exocentric (third-person, worlds-in-miniature [25] for example) and hybrid techniques (voodoo dolls [21] for example) are interesting, we are concerned with target acquisition in egocentric (first-person) views in this paper.

Ray-casting [8] is perhaps the simplest egocentric 3D selection technique, but performs poorly with small or distant targets [23]. This is due to the fact that small angular changes cause large movement at long distances. Virtual hand approaches, on the other hand, build on the metaphor of controlling an actual hand that allows for pointing and selecting objects using a grabbing operation [9]. To combat the problem of limited reach, the Go-Go interaction technique [22] allows for non-linear extension of the virtual arm of the user.

Andujar and Argelaguet [2] describes a technique for 3D virtual environments that decouples visual and motor space using a virtual representation of the user's preferred interaction space. However, they apply the concept to interaction with 2D applications projected onto surfaces integrated in the 3D environment, and not to 3D object picking itself.

The 3D bubble cursor [26], a volumetric extension of the 2D bubble cursor [14], is of particular relevance to this paper. The 3D bubble cursor dynamically resizes to always select the target closest to the pointer regardless of the actual position of the pointer. While clearly a highly efficient selection mechanism, it may provide "too much" assistance for some applications such as games where the user wants to feel fully in control.

3 Semantic Pointing in 3D

Semantic pointing [7] proactively utilizes information that is readily available in a graphical system but is typically never used: the location and size of the potential targets that the user can select and click. While the original concept was introduced for 2D user interface elements such as buttons, sliders, and window title bars, we adapt the idea to 3D object picking in this paper. Our adaptation is called *semantic 3D pointing*, and operates by essentially shrinking empty space and expanding target sizes in motor space for the purposes of pointing.

In the following subsections, we first describe the general principles of semantic pointing and then discuss its adaptation to 3D object picking. We then describe our current implementation. While the ideas we introduce apply to any relative input device, we will be directly referring to the mouse for the remainder of this text.

3.1 Decoupling Visual and Motor Space

In most graphical systems, motor and visual space are typically correlated for relative input devices—such as the mouse—using a constant scale factor. Thus, moving the mouse a certain distance (motor space) causes the pointer on the screen to move a corresponding distance (visual space). In such environments, it is possible to model the index of difficulty (*ID*) of selecting a target of size W at a distance D using Fitts' law [12]: $ID = \log_2(D/W + 1)$.

From an information theory point of view, the index of difficulty represents the amount of bits of information that the user gives the system. However, as argued by Blanch et al. [7] and Guiard et al. [15], while a screen may be as large as 1920×1200 pixels or more, the semantic meaning that the user is trying to convey with the pointing action is to select one target out of a small set of potential targets (also known by the system). In other words, the pointing task is often needlessly difficult.

Semantic pointing, then, changes the mapping between visual space and motor space depending on the local proximity to potential targets. In an empty area on the screen that is far from potential targets, the factor is changed to **speed up** movement in visual space, whereas in areas in close vicinity of targets, the movement in visual space is **slowed down** to allow for higher accuracy. A non-linear *scale function* is used to control the ratio between visual and motor space. As has been shown by Blanch et al. [7], this approach can significantly decrease the task time or errors for pointing.

3.2 Adaptation to 3D Object Picking

In a general 3D scene, we do not usually have the luxury of controlling the user's location and orientation. This means that we have no way of knowing in advance how potential targets will be projected onto the screen as the user navigates through the scene, nor how they will be deformed by perspective and occlusion by other targets and background scenery. In fact, while Blanch et al. deal with geometric primitives such as rectangles and points, our targets in a 3D context may be any shape or form.

Whenever the user moves the mouse in the physical world, the interaction technique must translate this movement into visual space. This requires that the underlying *semantic space* is known, i.e. the layout of potential 3D targets on the 2D screen space representing the application viewport. This space is queried at the current location of the pointer, yielding a distance measure to the nearest target. We use this measure as input to the scale function that controls the mapping between motor space and visual space.

3.2.1 Semantic Space Representation

We extract the semantic space of a 3D scene by drawing object identifiers to an *object buffer* at the same time that the scene itself is drawn. Scenery objects that should have no impact on pointing are assigned object identifier zero. While not as necessary for semantic pointing, assigning a unique identifier to each object (or an identifier that is unique to a certain class of objects) allows for the object buffer to be used as a pick buffer as well. Figure 3 shows a 3D scene and its corresponding semantic space.

A typical implementation would employ the standard *stencil buffer* as an object buffer. However, reading back the stencil buffer from video memory is a potentially expensive operation, so this operation should only be performed when necessary (i.e. for animated scenes or when the viewpoint is changed and the mouse pointer is moved).

To speed up spatial queries in the object buffer, we build a quadtree representation encoded in a linear array (Figure 2). The buffer dimension is expanded to the nearest multiple of two, with any unused space filled with empty semantic information (zero). This operation also only has to be done when the scene changes. Each node in the tree maintains a flag whether its four subtrees are empty or not, allowing for fast culling of whole branches while querying.

3.2.2 Querying Semantic Space

To derive the scale factor between visual and motor space, the semantic space must be queried for each mouse movement at the current location of the pointer. This is done using a level-order traversal of the quadtree representing the semantic space (see the previous section). Every empty subtree is discarded from further traversal, thus culling the search space.

Since an internal quadtree represents a region and not a specific location on the screen, a distance interval is computed for all such nodes (see Figure 2 for an example query for pointer position P). By keeping track of the current minimum and maximum possible distance for each level of the tree, we can quickly discard nodes whose minimum possible distance is further away than the level-wide maximum possible distance. Such nodes cannot end up containing the nearest target, and are thus out of the running.

Algorithms 1 and 2 give pseudocode listings for this spatial query algorithm. The two priority queues Q and NQ provide the current level and the next level's nodes to be visited, and they are sorted in ascending order by the minimum distance of each node.

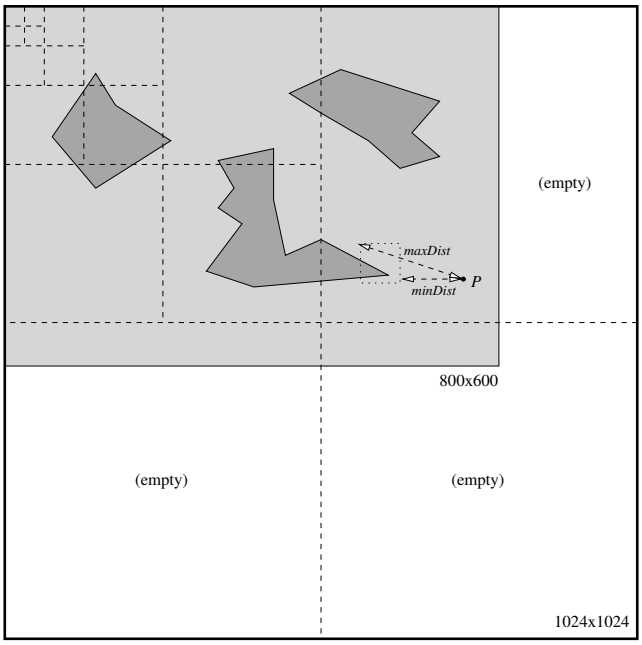

Figure 2: Quadtree representation of a semantic object buffer. The dotted rectangle shows an example spatial query for a node where the minimum and maximum possible distances from the pointer P are computed.

3.2.3 Scale Function

We have experimented with two different non-linear scale functions: one with a linear slope starting from a specific cutoff distance, and one non-linear curve (Figure 4). Both functions are available in our prototype implementation library and can be selected as appropriate depending on the needs of the application.

For a pointer at distance D from the nearest target, we define the linear scaling function $s_{lin}(D)$ as

$$s_{lin}(D) = \begin{cases} M & \text{if } C < D \\ \frac{D}{C}(M-m)+m & \text{if } 0 < D \leq C \\ m & \text{if } D = 0 \end{cases}$$

where M is the maximum mouse scale factor, m is the minimum, and C is the cutoff distance.

Accordingly, the inverse scale function $s_{inv}(D)$ is defined as

$$s_{inv}(D) = \frac{(m-M)}{(D+1)^S} + M$$

where M is the maximum mouse scale factor, m is the minimum, and S is a scaling constant (empirically derived).

3.2.4 Visual Feedback: Pointer Scaling and Color

Depending on the application, it can be advantageous to provide visual feedback to the user on the state of the semantic pointing functionality. For object picking in a CAD application, it is helpful to the user to know the current sensitivity of the mouse. We provide this by optionally also scaling the size of the mouse pointer using the scale function (Figure 5). This directly conveys the feeling that the mouse pointer is becoming smaller and more precise in the vicinity of potential targets. It has the added benefit of occluding a smaller part of the screen in important areas of interaction.

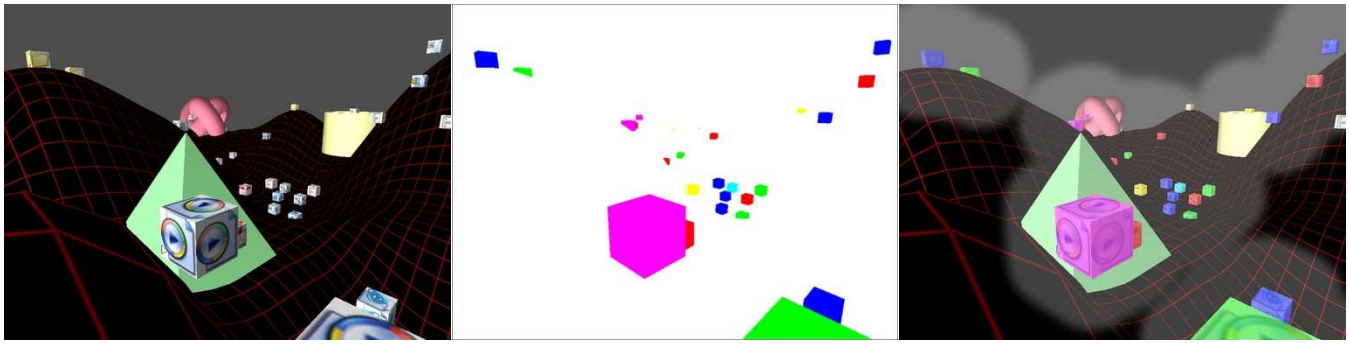

Figure 3: Standard framebuffer image (left) and semantic space image (middle) of a 3D scene. The right image shows the semantic space, including the local influence of targets, overlaid on the image space.

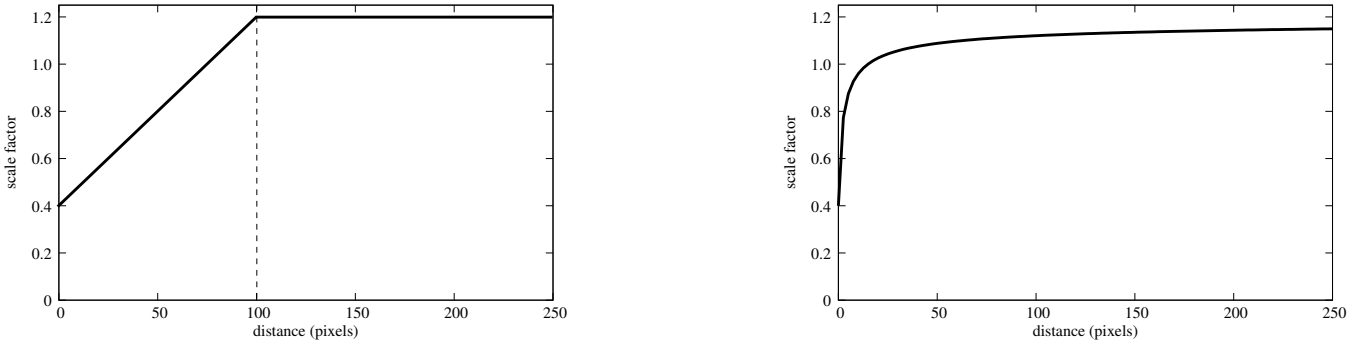

Figure 4: Linear (s_{lin}) and inverse (s_{inv}) scale functions.

We can also improve visual feedback further by highlighting the cursor in another color whenever the pointer is hovering directly over a potential target, or smoothly as a function of the distance to the nearest target. The same can be done for transparency, making the interior of the pointer (but not the outline) semi-transparent while hovering over targets to improve visibility.

Our visual feedback is simplistic in that it only conveys information on the semantic pointing functionality; more advanced mechanisms can dynamically enlarge cursors to directly improve object picking and selection [14, 17, 27].

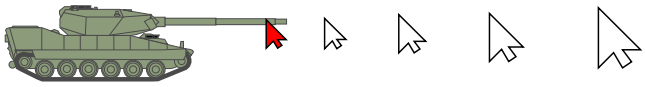

Figure 5: Pointer scaling and highlighting. As the pointer approaches and mouse sensitivity increases, the pointer shrinks accordingly. Hovering over the target highlights the cursor.

3.2.5 Bubble Cursor Extensions

Since our algorithm is able to derive the distance and position of the nearest pixel belonging to a target on the screen at any point in time, it is possible to extend the object picking mechanism to the bubble cursor [14]. In other words, instead of requiring that the user explicitly clicks on a pixel belonging to a target to pick the target, the currently closest target to the cursor can be highlighted and it is sufficient to click to pick it. Essentially, the cursor's activation area is expanded to always include the nearest target.

Furthermore, like for the bubble cursor and its volumetric 3D equivalent [26], we can provide further visual feedback on the currently selected object by explicitly drawing the radius of the current bubble. Figure 6 shows an example of the bubble selection space for a sample view of a 3D scene.

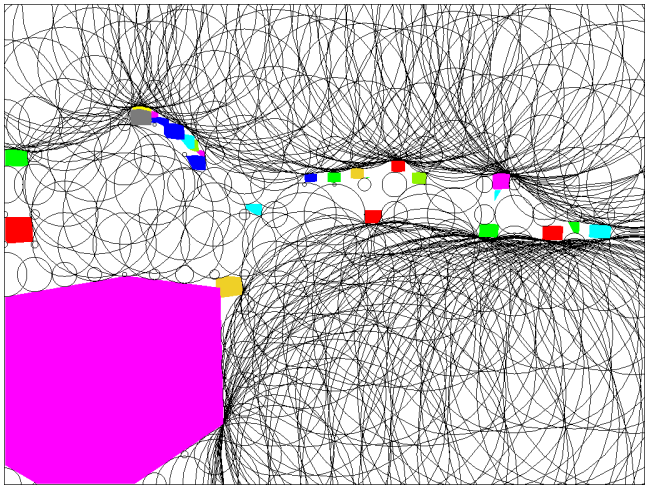

Figure 6: Bubble selection for the whole screen overlaid with the semantic space.

3.3 Implementation

We have implemented semantic 3D pointing as a prototype library using C++. The implementation uses OpenGL for rendering and

Algorithm 1: Semantic quadtree spatial query.

```
   Input: pointer position (x and y)
   Output: distance to nearest target d
1  dim ⟵ quadtree dimension
2  (minD, maxD) ⟵ getMinMaxDist(x, y, 0, 0, dim)
3  priority queue Q ⟵ ∅
4  priority queue NQ ⟵ (0, 0, dim, minD, maxD)
5  while NQ not empty do
6      Q ⟵ NQ
7      NQ ⟵ ∅
8      first ⟵ true
9      while Q not empty do
10         N ⟵ Q.pop()
11         if N.minD > maxD then
12             continue
13         if first then
14             minD ⟵ N.minD
15             maxD ⟵ N.maxD
16             first ⟵ false
17         else
18             minD ⟵ min(minD, N.minD)
19             maxD ⟵ min(maxD, N.maxD)
20         if N.dim = 1 then
21             return minD
22         checkChild(x, y, NWChild(N), NQ)
23         checkChild(x, y, NEChild(N), NQ)
24         checkChild(x, y, SWChild(N), NQ)
25         checkChild(x, y, SEChild(N), NQ)
```

Algorithm 2: checkChild

```
   Input: pointer position (x and y), child C, priority queue Q
1  if C is empty then
2      return
3  (min, max) ← getMinMaxDist(x, y, C.x, C.y, C.dim)
4  Q.push(C.x, C.y, C.dim, min, max)
```

SDL[1] as the underlying windowing system glue to get access to input and window operations. Furthermore, since we are controlling the motion of the mouse pointer, we cannot use the standard system pointer but instead must implement our own. SDL allows us exclusive access of the mouse and keyboard input devices as well as to turn off the default mouse cursor. This way, we are free to draw and manage our own mouse pointer.

Our implementation makes use of OpenGL and the stencil buffer as an object buffer for efficient rendering of both the 3D scene as well as the semantic space simultaneously. Semantic 3D pointing requires only a single bit in the stencil buffer (target or not), but we use 8 or 16 bits to allow for unique object identifiers to be stored in the buffer. This in turn allows us to use the stencil buffer as a pick buffer whenever the user actually clicks on the screen.

To further improve performance, it is possible to do partial updates of the stencil buffer for a small rectangular area around the current location of the mouse pointer (`glReadPixels` can optionally read back only a part of the buffer). The algorithm for building the distance quadtree can be similarly modified for partial updates. Since semantic 3D pointing only needs local distance information, this is a perfectly viable approach.

[1] *http://www.libsdl.org/*

4 Case Study: Usage in 3D Game Design

The main motivation for our work on semantic 3D pointing was for the new wave of 3D games where the player is expected to perform a great deal of interaction with the surrounding environment (as opposed to merely blowing it up). The real-time strategy game *Spring* (Figure 1) is an excellent example of such games.

It may be argued that for some games, the actual process of finding *which* targets are graspable is part of the challenge, and that semantic 3D pointing would defeat this purpose because it would give hints to the player. Whenever the player's pointer is in proximity of an object that can be interacted with, it slows down. However, we claim that the time for employing such artificial barriers in game design is long past, and that streamlining the user interface is as important for a computer game as it is for an office productivity application. [2] Furthermore, instead of relying on object obscurity in a scene (i.e. where part of the game challenge is to actually find the objects that can be interacted with), designers can instead introduce unimportant objects (so-called "red herrings") to complicate the game experience.

Semantic 3D pointing may have a use even for computer games or simulations where object picking is definitely a part of the challenge, such as for first-person shooter games where the user is actually aiming a gun using the mouse. For such games, semantic 3D pointing can help in this task for novice or beginner players who do not want to play the game on the hardest difficulty level; semantic 3D pointing can even provide a smooth level of support by changing the ratio between the maximum M and minimum m mouse sensitivity constants. It may also be a way to model that the game character itself has some innate skill at marksmanship.

Towards studying the use of the technique described in this paper in the context of computer games, we have developed two game-like 3D applications with semantic 3D pointing (Figure 7). One is based on a first-person shooter (FPS) in an indoor environment, the second is inspired by real-time strategy (RTS) games on a large outdoors battlefield. In both cases, we use the OpenGL for rendering and the stencil buffer to capture the screen space projections of all targets the user can interact with. The background scenery (walls, floor and ceiling for the FPS, the terrain for the RTS) is merely used for target occlusion purposes (filling the stencil buffer with object index zero). In contrast, distractor objects (furniture for the FPS game, trees and rocks for the RTS game) do have an impact on the semantic pointing space, but are not clickable.

5 Informal User Study

The basic benefit of semantic pointing has already been proven by Blanch et al. [7]. We submit that these results will transfer to the 3D adaptation, and thus we can focus our efforts on evaluation of the technique in the context of real applications of 3D picking. Towards this end, we make use of the two game-like 3D applications developed in the case study.

5.1 Participants

We recruited 8 unpaid participants (7 male, 1 female) from the graduate and undergraduate student population of our university. Ages ranged from 20 to 30 years. All participants had at least some previous experience in 3D computer games or applications, were not color blind, and had normal or corrected-to-normal vision.

[2] This is particularly true for online games such as Blizzard's *World of Warcraft*, where some players spend a comparable amount of time per day playing as they spend in Microsoft Word or Excel.

Figure 7: Screenshots from the two game-like applications with semantic 3D pointing (the left image shows the FPS, the right shows the RTS).

5.2 Procedure

The participants were briefly instructed in the motivation for the project and were allowed to practice using both scenarios until they felt comfortable using the technique. However, they were not initially told of the semantic 3D pointing technique and were just instructed to perform their task as fast as possible.

After this, the participants were exposed to the two scenarios. For each scenario, the task was to freely navigate and pick all of the targets of a specified type that the user could find in the environment. Picking a target (using a left-click) removed it from the environment (whereas picking a distractor had no effect). The participants had access to a simple overhead map in the upper-left corner of the display that showed the location of all remaining targets as well as their own position and view direction. In both scenarios there were a total of 20 targets to pick (different spatial configurations for each scenario).

- *FPS:* first-person shooter scenario in an indoor environment consisting of a one-floor building of interconnected rooms. The user's position was constrained to the floor. The target objects were giant locusts (left image in Figure 7).
- *RTS:* real-time strategy scenario in an outdoor battlefield consisting of rolling terrain and vegetation. The user's viewpoint was constrained to a horizontal plane high above the world (i.e. an "eye in the sky"). The target objects were tanks (right image in Figure 7).

Each participant performed task sets in both of these scenarios under the following two conditions:

- *Standard pointing:* standard mouse control with a constant scale factor between motor space and visual space.
- *Semantic 3D pointing:* full semantic 3D pointing functionality using the inverse scale function.

The participants received the conditions in counterbalanced order to prevent systematic effects of practice from affecting the results. The test platform silently collected task completion time and pointing accuracy for all conditions and scenarios. With 8 participants performing 2 scenarios under 2 different conditions, we collected 32 trials in total.

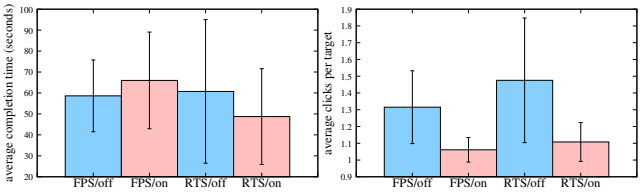

Figure 8: Timing and correctness results per scenario and condition.

For standard pointing, the scale factor was 1:1, so moving the mouse one unit in the physical world resulted in moving the pointer one pixel in visual space. For the semantic pointing condition, the maximum scale factor was 1:1.2 and the minimum was 1:0.4. These values were found using a pilot session.

5.3 Results

We collected quantitative performance measurements for all participants (Figure 8 shows a summary). In addition, we conducted extensive post-test interviews with two of our participants to try and gauge their subjective perception of the technique.

5.3.1 Performance

We analyzed our results using a one-way ANOVA, investigating the impact of semantic pointing on time and accuracy. For time performance, we found no significant difference ($p > .05$): 59.69 (s.d. 26.20) seconds without semantic 3D pointing versus 57.38 (s.d. 23.91) seconds with semantic 3D pointing. For accuracy, measured as the number of clicks per target (CPT), participants with semantic 3D pointing performed better than without: 1.40 (s.d. 0.306) clicks per target with no semantic 3D pointing compared to 1.08 (s.d. 0.097) with semantic 3D pointing. The latter is also a significant difference ($F_{1,30} = 15.032, p < 0.05$). Factorial 2×2 analysis, taking also scenario into account, shows the same results.

5.3.2 Interviews

For two of the participants, we conducted extensive post-test interviews. We first outlined the idea behind semantic 3D pointing and

demonstrated the operation of the technique, and then let the participants themselves perform the game scenarios again, but this time with awareness of the technique as well as with combinations of the visual feedback active (pointer scaling and highlighting).

The participants were very impressed with semantic 3D pointing for selecting targets in both scenarios. In general, they found more use for the technique in the RTS scenario, where the world was more open and semantic pointing helped them to acquire targets at long distances. As long as the participant was able to distinguish a target, the technique helped them to also pick it. However, participants found the technique useful even in the FPS scenario, particularly with visual feedback turned on.

Both participants appreciated the visual feedback, naming it superior to having no visual feedback. Both thought that pointer scaling provides exactly the right metaphor of the pointer becoming more precise. However, on the negative side, visually finding the pointer when it is lost is more difficult. Opinions were divided on pointer highlighting, with one participant saying it helped them in acquiring small targets, whereas the other thought it made no difference. Participants also disagreed on the utility of the bubble cursor extension, one saying that it helped visual tracking, whereas the other thought it was distracting.

One issue that was brought to light by both participants was the fact that distractors can cause slowdowns in interactions and even loss of visual tracking of the pointer (in one instance, the pointer was literally "stuck" in a tree in the RTS scenario). In both scenarios, only the background scenery (walls, ceiling and floor for the FPS scenario, the terrain and water for the RTS scenario) was not flagged as potential targets, the other objects beyond the actual targets were also included in the semantic space. However, including only the actual targets—locusts and tanks—in the semantic space would not constitute a realistic task.

Figure 9 shows mouse tracks from an interaction session for a participant picking targets using a fixed viewpoint in a 3D environment. The subject was instructed to sequentially click on each target for a duration of three minutes. This was repeated both with standard pointing and semantic 3D pointing. Our system captured cinametic logs of the pointer movement during each session. As can be seen from the two images, the mouse tracks of semantic 3D pointing exhibit less variation and is tightly grouped around the targets, whereas for standard pointing, there is a lot of overshooting and "circling" of the targets.

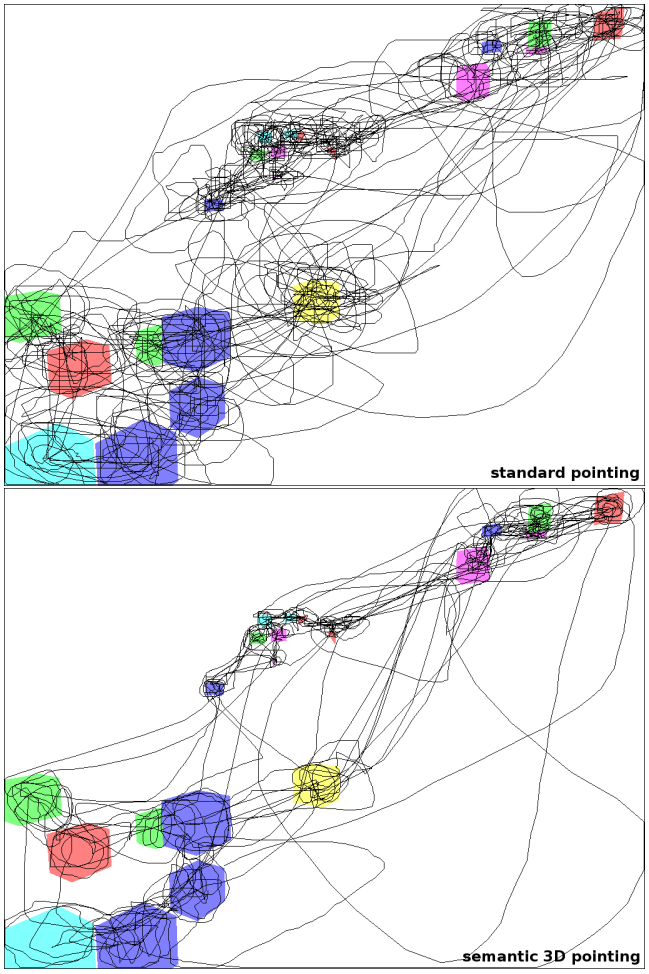

Figure 9: Comparison of mouse tracks with semantic 3D picking off (upper) and on (lower). Each image represents 3 minutes of activity. The semantic space is overlaid on top of each image.

6 Discussion

Our informal user study shows that accuracy is improved by semantic 3D pointing, but not task completion time. The accuracy improvement gives real motivation for the use of semantic 3D pointing in 3D applications since it clearly reduces the amount of erroneous mouse clicks performed by the user.

We believe the lack of a time difference is because the two game-like scenarios require a great deal of navigation, and this may overshadow any time improvements introduced by the interaction technique. At the same time, this navigation component is required for an ecologically valid evaluation of the technique in a realistic setting. Of course, semantic 3D pointing can be reduced to the original semantic pointing [7] in 2D if we fix the viewpoint, and thus the earlier findings on the utility of 2D semantic pointing should apply even for our 3D case.

Our adaptation of semantic pointing to 3D uses the distance to the nearest target on the screen as input to the scale function, whereas the original 2D semantic pointing by Blanch et al. [7] calculates a weighted scale by taking all of the targets on the screen into account. This is possible because they have a high-level model of the contents of the scene, whereas our approach is based on an bitmapped image of the semantic space. While it is certainly possible to derive such a high-level model even for a 3D scene, the object-buffer approach is simpler and yields an pixel-level accurate image of the semantic space even while the scene is being rendered.

That is not to say that the image-space approach is superior in all regards. Performance is an issue, because a typical screen consists of potentially millions of pixels. Simple floodfill approaches on a pixel-level will not be efficient enough for real-time purposes. Our quadtree representation and spatial query algorithm does provide real-time performance. Furthermore, the partial updates of the object buffer and distance quadtree proposed in Section 3.3 provide further optimizations. However, there may be even more efficient solutions to the problem.

Some interesting observations on potential idioms for using the technique were identified during the extended interviews for the user study. For instance, rather than going straight through a cluster of distractors to reach a target on the other side, participants would curve the pointer around the distractors to achieve higher movement speed and not get bogged down by the local influences of the distractors. One participant also used the bubble visual feedback

in helping visual tracking of the pointer when it was at its smallest size. These observations may indicate that performance for semantic 3D pointing is likely to increase further as users get fully comfortable with the technique.

Finally, it is interesting to note that in Blanch's original work [7], the study participants did not notice the modified mouse gain while still performing better with semantic pointing than without. We noticed the exact same effect: only 1 out of the 8 participants in the user study pinpointed a difference in the pointer movement between the two conditions, even though many participants felt that the task was "somehow easier" with semantic 3D pointing active. Of course, activating visual feedback would give away the game, but this is a decision that should be left to the designers.

7 Conclusions and Future Work

We have presented an approach to utilizing semantic pointing in 3D for improving object picking performance. The benefit of this work is to make interacting with entities in highly detailed and dynamic 3D environments easier in the face of increasing display space and more realistic computer imagery. Our adaptation uses an object buffer to extract the exact bitmapped projections of potential targets in the screen space. Using a hierarchical quadtree representation of this data, we are able to efficiently derive the distance to the closest target and modify the relative movement of the mouse pointer in visual space accordingly. An informal user evaluation studied object picking performance in two different game-like scenarios with and without semantic 3D pointing. Results from this study show promise for applying the technique to real 3D applications.

In the future, we anticipate targeting similar interaction problems with 3D games and simulations that arise from high-resolution displays and powerful graphics hardware. As discussed earlier in this paper, we believe that the actual interaction should never be a reason for a game being difficult, and that streamlined interaction design is instrumental for the new generation of computer games.

References

[1] D. Ahlström. Modeling and improving selection in cascading pull-down menus using Fitts' law, the steering law and force fields. In *Proceedings of the ACM CHI 2005 Conference on Human Factors in Computing Systems*, pages 61–70, 2005.

[2] C. Andujar and F. Argelaguet. Virtual pads: Decoupling motor space and visual space for flexible manipulation of 2D windows within VEs. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, pages 99–106, 2007.

[3] R. Balakrishnan. 'Beating' Fitts' law: virtual enhancements for pointing facilitation. *International Journal of Human Computer Studies*, 61(6):857–874, 2004.

[4] P. Baudisch, E. Cutrell, M. Czerwinski, D. C. Robbins, P. Tandler, B. B. Bederson, and A. Zierlinger. Drag-and-pop and drag-and-pick: Techniques for accessing remote screen content on touch- and pen-operated systems. In *Proceedings of INTERACT*, pages 57–64, 2003.

[5] P. Baudisch, E. Cutrell, K. Hinckley, and A. Eversole. Snap-and-go: helping users align objects without the modality of traditional snapping. In *Proceedings of the ACM CHI 2005 Conference on Human Factors in Computing Systems*, pages 301–310, 2005.

[6] B. B. Bederson. Fisheye menus. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 217–225, 2000.

[7] R. Blanch, Y. Guiard, and M. Beaudouin-Lafon. Semantic pointing: improving target acquisition with control-display ratio adaptation. In *Proceedings of the ACM CHI 2004 Conference on Human Factors in Computing Systems*, pages 519–526, 2004.

[8] R. A. Bolt. 'Put-that-there': Voice and gesture at the graphics interface. In *Computer Graphics (SIGGRAPH '80 Proceedings)*, volume 14, pages 262–270, 1980.

[9] D. A. Bowman and L. F. Hodges. An evaluation of techniques for grabbing and manipulating remote objects in immersive virtual environments. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, pages 35–38, 1997.

[10] A. Cockburn and P. Brock. Human on-line response to visual and motor target expansion. In *Proceedings of Graphics Interface*, pages 81–87, 2006.

[11] A. Cockburn and A. Firth. Improving the acquisition of small targets. In *Proceedings of the British HCI Conference*, pages 181–196, 2003.

[12] P. M. Fitts. The information capacity of the human motor system in controlling the amplitude of movement. *Journal of Experimental Psychology*, 47:381–391, 1954.

[13] G. W. Furnas. Generalized fisheye views. In *Proceedings of the ACM CHI'86 Conference on Human Factors in Computer Systems*, pages 16–23, 1986.

[14] T. Grossman and R. Balakrishnan. The bubble cursor: enhancing target acquisition by dynamic resizing of the cursor's activation area. In *Proceedings of the ACM CHI 2005 Conference on Human Factors in Computing Systems*, pages 281–290, 2005.

[15] Y. Guiard, R. Blanch, and M. Beaudoin-Lafon. Object pointing: a complement to bitmap pointing in GUIs. In *Proceedings of Graphics Interface*, pages 9–16, 2004.

[16] C. Gutwin. Improving focus targeting in interactive fisheye views. In *Proceedings of the ACM CHI 2002 Conference on Human Factors in Computing Systems*, pages 267–274, 2002.

[17] P. Kabbash and W. A. S. Buxton. The 'Prince' technique: Fitts' law and selection using area cursors. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 273–279, 1995.

[18] S. MacKenzie. Fitts' law as a research and design tool in human-computer interaction. *Human-Computer Interaction*, 7:91–139, 1992.

[19] S. Mackenzie and S. Riddersma. Effects of output display and control-display gain on human performance in interactive systems. *Behaviour and Information Technology*, 13(5):328–337, 1994.

[20] M. McGuffin and R. Balakrishnan. Acquisition of expanding targets. In *Proceedings of the ACM CHI 2002 Conference on Human Factors in Computing Systems*, pages 57–64, 2002.

[21] J. S. Pierce, B. C. Stearns, and R. Pausch. Voodoo dolls: seamless interaction at multiple scales in virtual environments. In *Proceedings of the ACM Symposium on Interactive 3D Graphics*, pages 141–145, 1999.

[22] I. Poupyrev, M. Billinghurst, S. Weghorst, and T. Ichikawa. The go-go interaction technique: Non-linear mapping for direct manipulation in VR. In *Proceedings of the ACM Symposium on User Interface Software and Technology*, pages 79–80, 1996.

[23] I. Poupyrev, S. Weghorst, M. Billinghurst, and T. Ichikawa. Egocentric object manipulation in virtual environments: Empirical evaluation of interaction techniques. In *Proceedings of Eurographics*, 1998.

[24] F. Steinicke and K. H. Hinrichs. Grab-and-throw metaphor: Adapting desktop-based interaction paradigms to Virtual Reality. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, pages 83–86, 2006.

[25] R. Stoakley, M. J. Conway, and R. Pausch. Virtual Reality on a WIM: Interactive worlds in miniature. In *Proceedings of the ACM Conference on Human Factors in Computing Systems*, pages 265–272, 1995.

[26] L. Vanacken, T. Grossman, and K. Coninx. Exploring the effects of environment density and target visibility on object selection in 3D virtual environments. In *Proceedings of the IEEE Symposium on 3D User Interfaces*, pages 115–122, 2007.

[27] A. Worden, N. Walker, K. Bharat, and S. Hudson. Making computers easier for older adults to use: Area cursors and sticky icons. In *Proceedings of the ACM CHI'97 Conference on Human Factors in Computing Systems*, pages 266–271, 1997.

[28] S. Zhai, S. Conversy, M. Beaudouin-Lafon, and Y. Guiard. Human on-line response to target expansion. In *Proceedings of the ACM CHI 2003 Conference on Human Factors in Computing Systems*, pages 177–184, 2003.