Symmetry Detection of 3D Objects*

Wookyoung Jung¹ and Takashi Yamauchi²

¹Department of Psychology, University of Illinois at Urbana-Champaign
jung43@illinois.edu

²Department of Psychology, Texas A&M University
takashi-yamauchi@tamu.edu

Developing realistic three-dimensional stimuli is an integral part of research on visual perception and cognition, including implicit and explicit forms of visual memory, symmetry perception, object recognition, and perceptual categorization. This is particularly important for object representation research because many theories that were developed with simplified two-dimensional stimuli turned out to be insufficient when tested with three-dimensional stimuli. In two experiments, we examined symmetry perception of naturalistic three-dimensional stimuli. The results indicate that symmetry perception for three-dimensional stimuli involves different processes from those employed for simple two-dimensional stimuli. Appendices include an archive of 3D stimuli created by 3D Studio Max (3ds Max) and a brief tutorial of the program geared for the construction of 3D stimuli for behavioral experiments.

Key words: Symmetry perception, three-dimensional stimuli, presentation duration, stimuli complexity, 3D stimulus creation, 3D Studio Max

1. Introduction

Forming coherent mental representations of visual objects is an integral part of the visual system (Peissig & Tarr, 2007). Such representations involve both perceptual and cognitive processes and provide essential information for understanding our visual environments (Palmeri & Gau-
Research on visual memory (Cooper & Schacter, 1992; Loftus, Miller, & Burns, 1978; Loftus & Palmer, 1974; Schacter & Cooper, 1993; Schacter et al., 1995; Squire, 1998), visual object classification (Mack, Gauthier, Sadr, & Palmeri, 2008), symmetry perception (Driver, Baylis, & Rafal, 1992; Yamauchi et al., 2006), and face recognition (Bukach, Gauthier, & Tarr, 2006; Gauthier & Bukach, 2007; McKone, Kanwisher, & Duchaine, 2006) are all linked to the central issues of object representation (Biederman & Bar, 1999; Biederman & Cooper, 1991, 1992; Marr, 1981; Marr & Nishihara, 1978; Peissig & Tarr, 2007; Poggio & Edelman, 1990; Tarr, 2003; Vuong & Tarr, 2004).

One technical problem blocking the progress of this line of research is the difficulty of developing realistic three-dimensional stimuli. A majority of visual perception and cognition studies have employed two-dimensional line drawings or simple pictures of three-dimensional objects as stimuli (Biederman & Bar, 1999; Biederman & Gerhardstein, 1993; Gauthier & Tarr, 1997b; Hayward & Tarr, 1997; Subramaniam, Biederman, & Madigan, 2000; but see Gauthier & Tarr, 1997a for some exceptions). The problem is that neurons in the visual cortex are highly interconnected, and, often times, these neurons respond differently in natural viewing conditions (Carandini et al., 2005, p. 10588; Felsen & Dan, 2005; Yuille & Kersten, 2006; Smyth, Willmore, Baker, Thompson, & Tolhurst, 2003; Touryan, Felsen, & Dan, 2005). In this regard, investigating behavioral responses to simple stimuli such as edges, blobs, bars, or line drawings may not give us the accurate picture of the visual system in question (Barlow, 2001; Felsen & Dan, 2005; Simoncelli & Olshausen, 2001).

This problem is particularly serious in symmetry perception research. Symmetries of objects are pivotal in both structural description models and view-based models of object representation. Research has shown that visual symmetries provide strong cues for image segmentation, such as distinguishing a figure from its background (Driver et al., 1992). The consensus is that symmetry perception occurs before and/or separately from the representation of local components and their configurations (Driver et al., 1992; but see Olibers & Helm, 1998). Evidence suggests that the complexity of local image features hardly interferes with the speed and accuracy of symmetry perception of two-dimensional figures (i.e., reflectional symmetry;
However, these claims were limited by results based on simplified two-dimensional stimuli (e.g., Baylis & Driver, 1994). Recently, although some progress was made in the field of symmetry perception with three-dimensional objects (e.g., Niimi & Yokosawa, 2009), most findings of symmetry perception have been performed with two-dimensional objects. To remedy this problem, we developed an archive of three-dimensional stimuli using 3D Studio Max (see Appendices A & B for stimulus archive and tutorial) and investigated the nature of symmetry perception with realistic three-dimensional stimuli (see Figure 1 for sample stimuli).

1.1 Symmetry and object representation

In Biederman’s (Biederman, 1987; Hummel & Biederman, 1992) recognition-by-components theory, the visual system extracts two-dimensional edge properties such as curvature, co-linearity, symmetry, parallelism, and/or co-termination from image data, and then constructs three-dimensional geometric primitives from these edge features (see also Layer 2 of “JIM” on p. 486, Hummel & Biederman, 1992). Similarly, in the Marr and Nishihara model (1978), axes of symmetry are computed in the service of generating a structural description of object shape. These models suggest that some form of symmetry detection occurs prior to the representation of local components and their structure. Consistent with these models, evidence suggests that symmetry perception can be achieved without the complete component-based structural descriptions of objects (Baylis & Driver, 1994; Yamauchi et al., 2006).

However, it is unknown whether the same process is involved in symmetry perception of realistic three-dimensional stimuli. Symmetry may emerge spontaneously without attention for simple two-dimensional stimuli (Driver et al., 1992 but see Olibers & Helm, 1998); yet, a more elaborate mechanism may be required for realistic three-dimensional objects. The construction of object structure by three-dimensional local components involves the binding of image features to objects, and this binding process is incomplete unless attention is given to the relevant locations (Logan, 1990; Treisman & Kanwisher, 1998; Stankiewicz, Hummel, & E. E. Cooper, 1998). This means that symmetry perception for three-dimensional objects can be an
incremental process, where symmetry is extracted gradually as the global structure of objects is recovered. We tested this idea in two experiments.

In Experiment 1, we examined the accuracy and response times of participants judging the symmetry or asymmetry of individual stimuli, where the complexity of stimuli and the duration of stimulus presentation were manipulated. We varied the number of local components of individual stimuli and examined how the complexity of stimuli, as measured by the number of distinct components that each object had, interacted with the accuracy and speed of symmetry perception. As observed in two-dimensional stimuli, if symmetry perception occurs spontaneously in realistic three-dimensional stimuli, the accuracy and response times of symmetry perception should be relatively independent of the duration of stimulus presentation and the complexity of stimuli. Contrary to this view, the results of Experiment 1 show that symmetry perception for three-dimensional stimuli is highly reliant on the duration of stimulus presentation and the complexity of stimuli. In Experiment 2, we implemented an incidental recognition memory experiment and examined whether stimulus complexity, as defined in Experiment 1, would also influence recognition memory performance. The results of Experiment 2 show that the stimulus complexity influences recognition memory performance when the study task involves incidental encoding of the global structures of objects.

2. Experiment 1

Experiment 1 examined the extent to which the presentation duration and the complexity of stimuli influence symmetry detection of three-dimensional objects. The hypothesis that symmetry perception for three-dimensional objects requires a gradual recovery of 3D structures predicts that presentation duration and stimulus complexity influence symmetry detection considerably. To test this prediction, participants judged the symmetry/asymmetry of stimuli in 5 different time durations (30, 40, 50, or 60 millisecond), as well as without time limitations. The complexity of stimuli was also manipulated by varying the number of distinct local components of which individual stimuli were composed.
2.1 Methods

2.1.1 Participants
All participants ($N = 213$) were Texas A & M undergraduate students who participated in this study for course credit. The participants were assigned randomly to one of five stimulus presentation duration conditions; (30ms, $n = 42$; 40ms, $n = 46$; 50ms, $n = 46$; 60ms, $n = 43$; unlimited exposure, $n = 36$). All participants had normal or corrected-to-normal vision.

2.1.2 Stimuli and procedure
Sixty symmetric objects and 60 asymmetric objects (Figure 1) were created by the 3ds Max program, release 6 [by Autodesk (Kinetix), San Rafael, California, USA]. To compose stimuli, we applied a number of modification techniques (described in detail in Appendix B), such as union, intersection, subtraction as well as lofting to 18 different kinds of primitive components (Figure 2). On average, our stimuli consisted of 4.6 components (4.9 and 4.3 components for symmetric and asymmetric objects respectively). The symmetric objects used in this study had a maximum of 11 components, and the asymmetric objects had a maximum of 10 components. The minimum numbers of components used in the symmetric and asymmetric objects were 2 and 1, respectively.$^1$

The stimuli extended approximately 4.0 degrees of visual angle.$^2$ The procedural details of stimulus generation are described in Appendices A and B; the tutorial is based on 3ds Max, release 10.

The task for participants was to indicate whether stimuli shown on the computer screen were symmetric or asymmetric by pressing designated keys. To start each trial, participants pressed the space bar, which generated a fixation cross, visible for 500 milliseconds. After the fixation cross disappeared, a stimulus was presented and remained at the center of the

---

$^1$ The “primitive components” used in our stimuli are defined and produced strictly in the 3ds Max program, and “primitives” are different from geometric primitives/parts as defined and characterized by Biederman (1987) or by Hoffman and Richards (1984).

$^2$ All stimuli and data for individual objects can be downloaded from http://people.tamu.edu/~takashi-yamauchi/stimuli/ and http://people.tamu.edu/~takashi-yamauchi/stimuli/data/
computer monitor until participants responded. To indicate the symmetry/asymmetry of the stimulus, participants pressed the left or right arrow key. Immediately after participants pressed one of the arrow keys, the monitor displayed a sign indicating a directive to press the space bar, which led to the presentation of the fixation cross and the next stimulus. The stimuli were presented by a Dell OptiPlex GX240 with Pentium IV processors (2.4 GHz) on a Gateway vx730 monitor with a resolution of 1280 x 1024 pixels.

2.1.3 Design
The experiment had a 5 (presentation duration; 30ms, 40ms, 50ms, 60ms, unlimited, between-subjects) x 2 (stimulus complexity; low, high) factorial design. The dependent measure was detection accuracy (d') and response times. For the detection accuracy measure, we defined “hit” as responding “symmetry” given symmetric objects, and “false alarm” as responding “symmetry” given asymmetric objects (see Barlow & Reeves, 1979; Wenderoth, 1997 for a similar procedure). Calculated this way, this measure helps control possible response bias (Ratcliff & McKoon, 1995; Roediger & McDermott, 1994). Because detection accuracy differed drastically in each presentation duration condition, the response times were measured for all trials rather than just the trials with correct responses.

To assess the impact of stimulus complexity, we measured the number of distinct local components that each stimuli had. This measure yielded two stimulus complexity categories: low and high complexity. For asymmetric objects, low complexity objects had 1 or 2 distinct kinds of components and high complexity objects had 3 to 5 distinct components (low complexity objects, n = 34; high complexity objects, n = 26 objects). For symmetric objects, low complexity objects had 1 or 2 distinct kinds of components and high complexity objects had 3 or 4 distinct kinds of components (low complexity objects, n = 33; high complexity objects, n = 27 objects).

2.2 Results
Detection accuracy (d'). The impact of presentation durations on sym-

---

3 Hit and false alarm scores equal to 1 or 0 were converted to 0.99 and 0.01, respectively, in order to calculate Z-scores.
Symmetries of 3D Objects

Symmetry detection is first reported and followed by the relationship between the stimulus complexity and detection accuracy. As shown in Figure 3, a 5 (presentation duration; 30ms, 40ms, 50ms, 60ms, unlimited) x 2 (complexity; high vs. low) ANOVA revealed that there was a significant main effect of presentation duration; \( F(4, 208) = 87.34, \text{MSE} = 0.04, p < 0.001, \eta^2 = 0.627 \). A contrast analysis applied to the 4 durations [the 30, 40, 50, 60 milliseconds presentation durations with contrast coefficients of (-3, -1, 1, 3)] reveals that there is a significant linear trend between the presentation duration factor and detection accuracy (\( d' \)), \( t(173) = 7.38, p < 0.001 \). The presentation duration factor did not influence the response time measure for both symmetric and asymmetric objects; \( F's < 1.0 \).

There was also a reliable main effect of stimulus complexity; \( F(1, 208) = 23.80, \text{MSE} = 0.186, p < 0.001, \eta^2 = 0.103 \), indicating that participants were more accurate for low-complexity stimuli than high complexity stimuli. The interaction between complexity and presentation duration was also significant; \( F(4, 208) = 2.64, p < 0.035, \eta^2 = 0.048 \). Planned comparisons suggest that symmetries of low complexity stimuli were detected more accurately than those of high complexity stimuli in the 60ms presentation duration and in the unlimited presentation duration; 60ms, \( t(42) = 3.76, p < 0.01 \); unlimited, \( t(35) = 3.68, p < 0.01 \) (Figure 3a).

Response time. The complexity factor also influenced the response time measure for symmetric, but not asymmetric, objects. The average response time for high complexity stimuli was longer than low complexity stimuli in symmetric objects; \( F(1, 208) = 14.96, \text{MSE} = 19239.7, p < 0.001, \eta^2 = 0.067 \), and the interaction between presentation duration and complexity was also significant; \( F(4, 208) = 3.65, \text{MSE} = 19239.7, p < 0.01, \eta^2 = 0.066 \). This interaction effect stemmed exclusively from the unlimited time exposure condition, \( t(35) = 6.84, p < 0.01 \). The complexity factor did not influence the response times of asymmetric objects; \( F(1, 208) = 1.48, \text{MSE} = 58798.0, p = 0.225, \eta^2 = 0.007 \). Complexity x duration; \( F(4, 208) = 1.87, \text{MSE} = 58798.0, p = 0.12, \eta^2 = 0.035 \).

Taken together, these results show that symmetry detection for three-dimensional objects depends on the duration of stimulus presentation and the complexity of stimuli, suggesting that detection of symmetries of 3D objects is unlikely to be spontaneous and preattentive.
3. Experiment 2

In Experiment 2, we investigated the relationship between symmetry perception and “stimulus complexity” indirectly in a recognition memory test. Previous studies investigating implicit memory of three-dimensional objects suggest that global structures of objects such as orientation axes of symmetry can be formed after an encoding task in which participants judged the left-right orientation of objects (Liu & Cooper, 2001; Yamauchi et al., 2006). This encoding task is also known to produce priming in symmetry/asymmetry judgments. Thus, if the stimulus complexity that is defined in Experiment 1 is psychologically real, it should affect the performance for the incidental recognition memory task. We tested this idea.

The experiment consisted of two phases: a study phase and a test phase. During the study phase, participants studied left-right orientations of 30 symmetric and 30 asymmetric objects. After the study phase they were given a recognition task in the test phase. This experiment was conducted incidentally in that participants were not aware of the intent of the experiment during the study phase (an incidental test). In an additional follow-up study, we also investigated the relationship between stimulus complexity and intentional recognition memory performance. The intent of this follow-up study is explained in the Result section.

3.1 Methods

3.1.1 Participants

A total of 60 undergraduate students (female = 40; male = 20) from Texas A & M University participated in Experiment 2 for course credit. All participants had normal or corrected-to-normal visual acuity.

3.1.2 Stimuli and procedure

The materials used in this experiment were identical to those described in Experiment 1. Of the total 60 symmetric objects and 60 asymmetric objects, 30 asymmetric objects and 30 symmetric objects were shown in the study phase, and all 120 (60 symmetric and 60 asymmetric objects) were shown in the test phase. We developed two versions of stimuli for counterbalanc-
In the study phase, 30 symmetric objects and 30 asymmetric objects were randomly presented one at a time at the centre of the screen for five seconds each. Participants examined each object as a whole for the entire 5 seconds and judged whether the object faced primarily to the right or to the left by pressing one of two specified keys (see Schacter et al., 1990, for this task). No mention was made at this point about the subsequent recognition task. Shortly after the study phase, participants proceeded to a recognition task, in which participants made ‘yes/no’ recognition responses for studied and non-studied objects.

3.1.3 Design

The main dependent measure was d’ calculated from hit and false alarm scores obtained in the recognition memory task. “Hit” is defined as the probability of responding “old” given that the stimulus appeared during the study phase (p(response = “old” | old)) and “false alarm” is defined as the probability of responding “old” given that the stimulus did not appear during the study phase (p(response = “old” | new). To analyze the relationship between stimulus complexity and recognition memory, an item-based linear regression analysis was applied to the data with the stimulus complexity measures (i.e., the number of distinct local components) as a predictor variable.

3.2 Results

Table 1 summarizes the results from the recognition memory test. The average hit minus false alarm scores of asymmetric objects and symmetric objects were 0.24 and 0.26, respectively. In both cases, these scores were significantly higher than a chance level, indicating that participants were able to distinguish individual objects in their recognition memory; asymmetric objects, $t(59) = 12.7, p < 0.001$; symmetric objects, $t(59) = 13.0, p < 0.001$.

As predicted, a linear regression analysis applied to individual objects revealed that there is a significant linear relationship between stimulus complexity and recognition memory performance when the study task was given incidentally; $F(1, 118) = 11.87, MSE = 0.02, p < 0.01; \beta = -0.30, p <$
4. Follow-up Test

In a follow-up study ($N = 30$), we also tested the effect of stimulus complexity in a recognition memory experiment in which participants were instructed to study individual objects carefully for the subsequent memory test (i.e., an intentional recognition memory task). The purpose of this follow-up study was to check the possibility that the effect of stimulus complexity stems from the left-right orientation task in which axes of symmetry were encoded, rather than the member encoding in general.

4.1 Methods

4.1.1 Participants

A total of 30 undergraduate students from Texas A & M University participated in Experiment 3 for course credit. All participants had normal or corrected-to-normal visual acuity.
4.1.2 Stimuli and Procedure
The materials used in this experiment were identical to those described in Experiment 2. Participants in this follow-up study did not carry out the left-right orientation task for individual stimuli during the study phase; instead, participants actively studied stimuli for a subsequent memory task in any way they wanted. Studies showed that the intentional memory task leads people to form semantic associations of stimuli (Cooper, Schacter, Ballesteros, & Moore, 1992; Schacter et al., 1990; Schacter & Cooper, 1993). In this regard, it was expected that the stimulus complexity as measured by the number of distinct components would not relate to participants’ recognition performance in this case.

4.1.3 Design
The main dependent measure was the same as Experiment 2.

4.2 Results
As expected, there was no statistically significant link between the stimulus complexity variable and the recognition memory performance in the intentional memory task; $F < 1.0$.

5. General Discussion
The results from these two experiments indicate that there is a strong possibility that symmetry perception for three-dimensional objects is an incremental process, where symmetry is extracted gradually as the global structure of objects is recovered. In Experiment 1, we reasoned that if symmetry perception emerges gradually as global three-dimensional information is recovered from the image data then symmetry detection should be influenced by the complexity and presentation durations of stimuli. The results of Experiment 1 were consistent with this view. Experiment 2 employed an incidental memory task. In this setting, participants judged the left-right orientation of stimulus objects during the study phase and later tested their recognition memory of these objects in the test phase. As predicted, the results of Experiment 2 revealed that recognition memory performance was highly correlated with stimulus complexity when the study task involved
incidental left-right orientation judgments, but not when the study task involved intentional encoding. Taken together, these results are consistent with the view that symmetry perception in three-dimensional objects emerges in an incremental process, where symmetry is extracted gradually as the global structure of objects is recovered.

5.1 Implications: Symmetry Perception and Object Representation

Clearly, at the very early stage of object representation, some form of part decomposition and the assignment of edges to a figure should occur. At this stage, some form of coarse descriptions of object parts occurs (e.g., Hoffman & Ridchards, 1984), and this coarse representation of object parts probably suffices for initial symmetry detection relevant to two-dimensional images. After this stage, symmetry perception for three-dimensional stimuli arises gradually as figure-ground segregation is achieved (see also Sekuler & Palmer, 1992; van Lier, Leeuwenberg, & Helm, 1997). We suggest that symmetry detection is likely to go through multiple-stages (Wagemans, 1997). Given the current finding that symmetry perception for three-dimensional stimuli depends on stimulus complexity and presentation durations, the later stages of symmetry perception may employ a constellation of features as cues determining the axis of symmetry, which can be identified, for example, by processing local energy functions (the intensity of pixels) and then local maxima of a convoluted image. This process proceeds iteratively as more fine-tuned features are recovered from the original images (e.g., Scognamillo, Rhodes, Morrone, & Burr, 2003).

In Biederman and Hummel’s model of object representation and recognition (Biederman, 1987; Hummel & Biederman, 1992; see also Marr, 1981; Marr & Nishihara, 1978), a limited number of geometric primitives and their configurations render the representation of a variety of basic-level objects. The model suggests that constructing three-dimensional local components from two-dimensional non-accidental image data requires the detection and integration of different image features (Biederman, 1987). In order to combine these properties, some attentional mechanisms may be deployed. We suggest that symmetry perception for three-dimensional objects appear as these local three dimensional features are recovered gradually. Future research should investigate exactly how the recovery of three-
Symmetries of 3D Objects

dimensional features eventually gives rise to the extraction of global axes of symmetry.

References


Vuong, Q. C., & Tarr, M. J. 2004. Rotation direction affects object recognition.
Symmetries of 3D Objects


Appendix A

Stimulus archive

Using 3ds Max software, we created 120 discrete objects (60 symmetric and 60 asymmetric objects) and their 1680 rotational variations. Individual objects were rotated by an increment of 30 degrees up to 180 degrees, and the rotation was made by two different coordinate systems, which roughly correspond to object-centered and viewer-centered coordinate systems (120 objects x 7 different angles x 2 coordinate systems = 1680 objects; see the Transforming Objects subsection in the Tutorial section of Appendix B regarding the details of reference coordinate systems). These objects were stored in jpg, bitmap, and 3ds max file formats. Thus, in total we produced 5040 files (see Figure 1 for 20 samples of our stimuli). We produced both symmetric and asymmetric objects in order to address the central role that visual symmetry plays in object perception. Symmetries often evoke perceived “goodness” of figures; they facilitate figure-ground segregation, and are detected spontaneously (Barlow & Reeves, 1979; Baylis & Driver, 1994; Corballis & Roldan, 1974; Driver, Baylis, & Rafal, 1992; Palmer, 1991; Palmer & Hemenway, 1978; van der Helm & Leeuwenberg, 1996; Wagemans, 1995). Symmetry also plays a crucial role in computational models of object representation and recognition (Biederman, 1987; Hummel & Biederman, 1992; Marr, 1981; Marr & Nishihara, 1978; Tarr, 1995; Ullman, 1996, 1998; Vetter & Poggio, 2002).

To compose stimuli, we applied a number of modification techniques (described in detail in the next tutorial section) such as union, intersection, subtraction as well as lofting to 18 different kinds of primitive components (Figure 2). On average, our stimuli consist of approximately 4.6 components (4.9 and 4.3 components for symmetric and asymmetric objects respectively). The symmetric objects used in this study contain a maximum of 11 components, and the asymmetric objects contain a maximum of 10 components. The minimum numbers of components used in the symmetric and asymmetric objects are 2 and 1, respectively.

One advantage of creating 3D objects by combining primitive components is productivity (Biederman, 1987). By selecting 4 components
randomly from 18 primitives, for example, $18^4 = 104,976$ objects can be produced relatively easily. Using 3ds Max, manipulating the size, color, orientation, and/or texture of the objects is straightforward. Thus, our
Figure 2. Eighteen main primitive components composing our 60 symmetric and 60 asymmetric objects.
Figure 3. The main results of Experiment 1: (a) symmetry detection accuracy (d’), (b) response times for asymmetric objects, and (c) response times for symmetric objects. The size of the error bars represents two standard error units.
stimuli can be applied and extended to a variety of studies addressing recent controversies regarding face recognition, visual object representation, and memory systems. The next section introduces a brief tutorial of 3d Max.

Appendix B

**Tutorial: Creating 3D stimuli with 3ds Max**

Recently, researchers studying visual perception and cognition have also employed 3ds Max (e.g., Vuong & Tarr, 2006). Because this software helps create a variety of three-dimensional stimuli with many manipulations, researchers of visual perception and cognition will benefit enormously by integrating this software into their research projects.

This tutorial consists of the following three topics: creating objects, transforming objects, and modifying objects. These topics cover essential skills necessary to build new object stimuli. We first briefly describe the main interface and then explain the major functions of 3ds Max.

**The Interface.** The interface of 3ds Max consists of the main toolbar, the view port, and the command panel (Figure 4). The main toolbar provides quick access to tools and dialog boxes for many of the regular tasks in 3ds Max. Included on the main toolbar are tools for selection, transformation (movement, rotation, and scale), undo/redo and rendering.

The view port displays the actual object being created. The default setup is a four-view layout, with three orthographic views (top, front, and left) and one perspective view. An orthographic view portrays all objects in actual size relative to one another, while a perspective view shows how objects would look in the real world. Orthographic views are essential for modeling because objects are not distorted by distance. When the view port is active, the rectangle border turns yellow.

The command panel is the control center for nearly every operation in 3ds Max. It is divided into six sections by tabs across the top, and all of the creation and modification tools are found in this panel. The six tabs featured on the command panel are Create, Modify, Hierarchy, Motion, Display, and Utilities (Figure 5). The next section will explain how these functions are used.
Creating Objects

Geometry. The creation of all default objects in 3ds Max starts with the create panel (Figure 4). This panel is located at the first position of the command panel, indicated by an arrow icon pointing to a star. The create panel is separated into two parts: Geometry and Shapes. 3ds Max offers a list of default geometric primitives. One category of primitives is standard primitives, which includes Box, Cone, Sphere, Geosphere, Cylinder, Tube, Torus, Pyramid, Teapot, and Plane (Figure 6a). Another category is extended primitives, which includes Hedra, ChamferBox, OilTank, Gengon, RingWave, Prism, Torus Knot, ChamferCyl, Capsule, Spindle, L-Exit, Hose, and C-Exit (Figure 6b).

To create an object, you simply click the button for the primitive that you want to create and click on one of the viewports. When an object button is selected, it turns dark yellow, indicating that you are in the creation mode.
As you move your mouse, the primitive will appear in the view port (Figure 5). To change the color of the object, click the Object Color button on the command panel, and you can specify the red, blue, green, hue, saturation, and whiteness values of the object. We set these values 170, 170, 0, 0, 0, and 170, respectively, in all our examples (Figure 1).

**Shapes.** Shapes are methods used for 2D object creation, while Geometry is for 3D object creation. Shapes are composed of Splines, NURBS Curves, and Extended Splines. To begin, Splines include Circle, Arc, NGon, Text, Section, Rectangle, Ellipse, Donut, and Helix. The procedure for creating shapes is identical to that of creating geometry.

![Shapes](image)
Figure 6. (a) Standard primitives used in creating default objects: Box, Cone, Sphere, Geosphere, Cylinder, Tube, Torus, Pyramid, Teapot, and Plane. (b) Extended primitives used in creating default objects: Hedra, ChamferBox, Oil Tank, GenGon, RingWave, Prism, Torus Knot, ChamferCyl, Capsule, Spindle, L-Exit, Hose and C-Exit.
Transforming Objects – Moving, Rotating, and Scaling

A transformation is defined as a change in position, rotation, or scale of an object. Transformations are different from modifications in that modifications change the object’s geometry, but transformations do not. The three transform buttons are located on the main toolbar (Figure 7). Using these buttons, objects can be selected and transformed by using the mouse to drag in one of the viewports. The orientation of transformation can be defined by one of nine systems of reference axes (view, screen, world, parent, local, gimbal, grid, working, and pick). These reference coordinate systems define the origins and directions of transformation. We will explain screen and local coordinate systems because they are most relevant for our purpose. The screen coordinate system is defined by the horizontal (x-axis), vertical (y-axis) and depth (z-axis) coordinates of the computer screen you are working on (Figure 8a). Because the screen coordinate system relates to the position of the viewer, this coordinate system is a “viewer-centered” system (Marr & Nishihara, 1978). The local coordinate system is defined by the “pivot point” (center) of the selected object (Figure 8b & c). For an object created from multiple primitives, the pivot point is the average of all its primitive components (Figure 8d). In our stimulus archive, we included 1680 sample objects that were rotated by the screen and local coordinate systems (120 symmetric and asymmetric objects were rotated 7 times from 0 to 180 degrees in an increment of 30 degrees; this was done both in the local and screen coordinate systems).

Figure 7. Three transform buttons of the main toolbar located in the command panel: The move transform is for changing position, the rotate transform is for changing viewpoint, and the scale transform is for changing size.
Move. An object can be moved in any of the three directions along the x, y, or z coordinate. To move an object, you click on the Select and Move button on the main toolbar (Figure 7). Next, select the object you wish to move and drag the object in the viewport to the desired location. Translations are measured in the defined system, which may be inches, centimeters, or meters.

Rotate. Rotation is the process of spinning the object about its Transform Center. To rotate objects, click the Select and Rotate button on the main toolbar, select the object to rotate, and drag it on a viewport. Rotations can

Figure 8. Illustrations of the screen coordinate system: (a) the local coordinate system (b), (c), and (d). The screen coordinate system defines the axes of transformation with respect of the screen, and the local coordinate system defines the origin of the axes by the center of the object: (b) and (c). For a combined object, the origin is the average of all its primitive components, (d).
be mechanically specified in degrees as well as created manually by moving the mouse.

**Creating Compound Objects**

To create complex three-dimensional objects, we can combine lines and shapes as well as geometric primitives such as Box, Cone, Capsule, Cylinder, Hedra, Hose Tube, and Torus. Using the Compounding Objects tool, which can be found in the drop-down menu in the command panel (Figure 9. Three types of Boolean operations used in creating complex objects: Union, Intersection, and Subtraction. In Union, the Boolean object contains the volume of both original objects. The overlapping portion of the two original objects is occluded. In Intersection, the resulting object includes only the point where the original objects overlap. And in Subtraction, the Boolean object contains the volume of one original object with the intersection volume subtracted from it.)
9), complex objects can be created from the basic primitives. The Compounding Objects tool includes Morph, Conform, BlopMesh, Boolean, Loft, ProBoolean, Scatter, Connect, ShapeMerge, Terrain, Mesher, and Procutter. The following are explanations of the Boolean and Loft operations, which are most useful for creating three dimensional stimuli for psychological experiments.

**Boolean Operations.** Boolean Compounding combines two objects by performing a Boolean operation on them. In 3ds Max, a Boolean object is made from two overlapping objects. Boolean operations consist of three sub-functions: Union, Intersection, and Subtraction (Figure 9). In Union, the Boolean object contains the volume of both original objects, and the overlapping portion of the two original objects is occluded. The Intersection sub-function is used when the Boolean object contains only the portion that was common to both original objects. In Subtraction, the Boolean object contains the volume of one original object, minus the intersection volume. We will demonstrate this process using the example of combining Sphere and Box (Figure 7).

To begin, arrange two primitives (e.g., the sphere and the box in Figure 9) that you want to combine on the viewport. Select a target primitive (e.g., the spear in Figure 9) using the Select Object button on the Main Toolbar. On the create panel, click on Geometry and select Compound Objects from the scroll menu. Click on the Boolean button of the object type and then drag the mouse over an empty area. The mouse cursor will change to a hand. Press the mouse button and drag it to move the panel upward. Now, a portion of the panel that was previously hidden will be visible. Next, we select a suitable Boolean Operation. For this operation, we clicked on the Union radio button. The original primitive that was selected earlier (the sphere in Figure 9) is defined as “Operand A.” Now, click on the button “Pick Operand B” and click the primitive (e.g., the box in Figure 9) that you want to be combined. A Boolean operation will be performed for the two objects, and a new Compound Object will be created. Examples of objects we made using the Boolean operations are shown in Figure 10.

**Loft operations.** The loft operations turn a 2D shape into a 3D form. For example, through compounding a circle and a line, a tubular shape is created. That is, a circle is lofted along a path to construct a tubular shape.
Figure 10. Sample stimuli created by the Boolean functions in 3ds Max.
Figure 11. Example figures of the loft operation. A curved line that will act as the path and a star drawn in the top view.

To use this operation, create a curved line that will act as your path. Then, create a shape (Figure 11). Using the Geometry button, choose Compound Objects. Click on your path then click the Loft button under the Object Type heading. Press the Get Shape button under the Creation Method head-
Wookyoung Jung and Takashi Yamauchi

Rendering Objects

The process of rendering creates two-dimensional pictures of three-dimensional objects. A rendered object shows the final image of an object that can be used as a stimulus. When rendering, the perspective view shows how objects will look in the real world. Therefore, the process of rendering is operated in the perspective view.

To begin, click on the Perspective View button and make the viewport active. When Rendering from the main menu bar is selected, Render and Environment will appear as two options: select Render. Select the desired output size in the common parameters. For the sample objects shown in Figure 10, we selected a resolution size of 320 × 240. Next, select Environment and Ambient (the ambient value is necessary to control global lighting). We set the Ambient values of Ambient, where the value of red, green, blue, hue, sat, and whiteness at 130, 130, 130, 0, 0, and 130, respectively.

![Environment and Effect setup](image)

**Figure 12.** Setting up Environment and Effect in rendering the created object. In the Environment, Ambient is composed of six factors: Red, Green, Blue, Hue, Sat, and Value. Each Value was set up to 130, 130, 130, 0, 0, and 130, respectively. Ambient is necessary to control the global lighting of the created object.

ing and click your shape. A compound loft object will appear (Figure 11).
Symmetries of 3D Objects

(Figure 12). The ambient value is necessary to control global lighting. Next to Environment is the Effect tab. Using this tab, click on Add, and then select Brightness and Contrast, which should be set to 0.5 and 0.5. The last step in rendering is to save the objects. After each value of rendering is set, click on the Render button, which allows the created objects to be seen (Figure 13). When saving, select the Bitmap image file and 8 bit Optimized palette (256 colors).

**Archiving files**

3ds Max scenes can be archived using the “File > Archive” Menu command. Archive creates compressed archive files, or text files, listing all relevant files and their paths. Archiving a scene ensures that all scene files necessary for the complete scene are bundled along with the main scene file. This is especially useful if you need to send the file to another computer.
In summary, the general steps to create three-dimensional objects are as follows: (1) select the default primitives you would like to include in your object from the command panel; (2) drag the primitives to the viewport; (3) modify them (i.e., move, rotate, or scale) as needed using the Transformation menu on the main tool bar; (4) combine the primitives using Boolean or Loft operations (i.e., Union, Intersection or Subtraction) shown in the Compounding Objects tool; (5) render the objects with suitable color, ambience, and lighting; (6) save the objects in an appropriate format (jpg, bitmap, gif, or pdf).