Paradoxical Figures That Adapt to the Viewer's Relative Position

Matías Gárate[1] and Paul Weber[2]

[1] Fennange, Luxembourg; matiasgarate3d@gmail.com
[2] whywedo., Fennange, Luxembourg; paul.weber@whywedo.com

Abstract

We present a 3D modeling algorithm that takes into account the orientation of a virtual camera to build impossible figures that adapt dynamically to the viewing angle. The parametric nature of this algorithm allows the viewer to observe these paradoxical objects from multiple angles in turn-around camera animations, and opens the possibility to create immersive experiences, where the audience can physically interact with artworks of impossible objects.

Introduction

Impossible objects depicted in the artworks of M.C. Escher, O. Reutersvärd, and Sandro del Prete have captivated the curiosity of mathematicians, artists, and the general public for decades. Typically these paradoxical figures cannot be built as 3-dimensional objects in the real world, or at least, not without the use of perspective tricks that mislead the viewer's perception of space, such as making the intended figure visible only from a single viewing angle (see 3D printed example in Figure 1). In 3D software however, it is possible to parameterize these 'perspective tricks' according to the orientation of a virtual camera, and thus to create an impossible figure that adapts itself to the viewing angle.

In this paper we present two parametric algorithms to create 3D mesh models of the 'Necker Cube' and the 'Impossible (mono)Bar' (Figure 2), and implement them using the software Blender [1]. We demonstrate how these figures can be rendered from multiple angles in turn-around camera animations, and finally propose how to create an interactive paradox, in the form of a Processing application [3], that adapts to the relative position of the viewer.

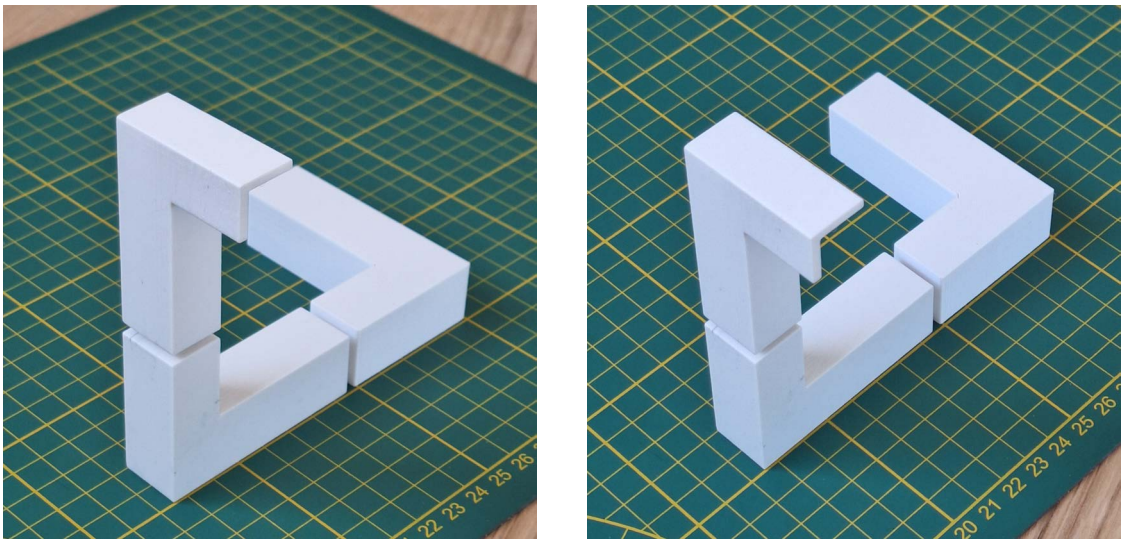
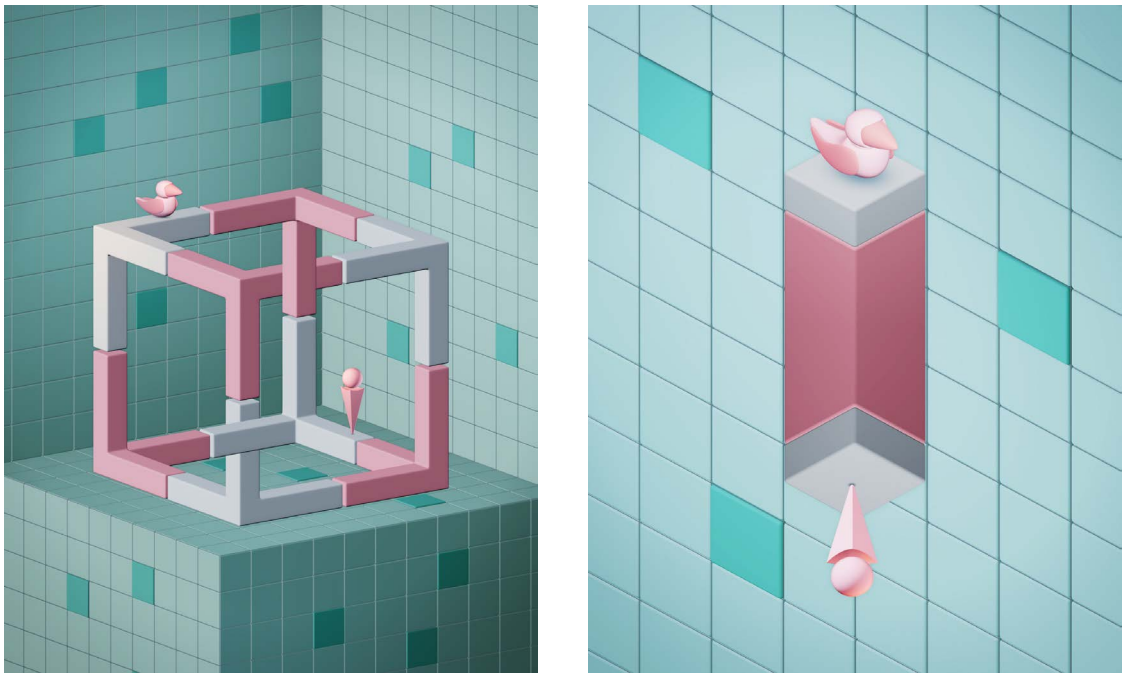


Figure 1: *3D printed variation of the Penrose Triangle [6], viewed from two different orientations.*



(a) **(b)**

Figure 2: *Artistic renderings of (a) the Necker Cube and (b) the Impossible Bar, both made in Blender.*

Coordinate Systems

To define the frame of reference, consider the 3D canonical coordinate system with the standard vector basis $\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}$. Now, consider the coordinate system of an orthographic (trimetric) camera (i.e. the camera that we use to render our image), which is oriented according to the co-latitude and azimuthal angles θ, ϕ as indicated in Figure 3(a).

The basis vectors of the 'Camera' system $\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}$ can then be written in terms of canonical basis through the following operation:

$$[\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}] = R_{\mathrm{z}}(\phi) R_{\mathrm{x}}(\theta) [\hat{\mathbf{x}}, \hat{\mathbf{y}}, \hat{\mathbf{z}}], \tag{1}$$

where R_i is the rotation matrix around the i axis. Solving the operation explicitly, we find:

$$\hat{\mathbf{u}} = \begin{bmatrix} \cos\phi \\ \sin\phi \\ 0 \end{bmatrix}, \; \hat{\mathbf{v}} = \begin{bmatrix} -\sin\phi\cos\theta \\ \cos\phi\cos\theta \\ \sin\theta \end{bmatrix}, \; \hat{\mathbf{w}} = \begin{bmatrix} \sin\phi\sin\theta \\ -\cos\phi\sin\theta \\ \cos\theta \end{bmatrix}, \tag{2}$$

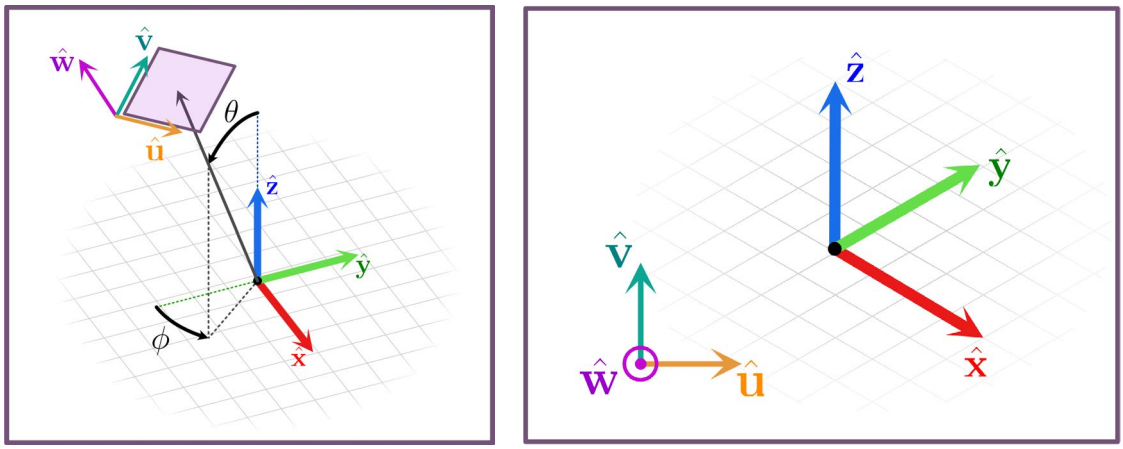
where the vectors $\hat{\mathbf{u}}, \hat{\mathbf{v}}$ correspond the horizontal and vertical axes of the rendered image, and the vector $\hat{\mathbf{w}}$ indicates the direction normal to the image plane, as shown in Figure 3(b).

Finally, it is also useful to define the projection of a 3D vector $\vec{a}$ on the 2D rendered image through:

$$\vec{a}_{2\mathrm{D}} = \begin{bmatrix} \vec{a} \cdot \hat{\mathbf{u}} \\ \vec{a} \cdot \hat{\mathbf{v}} \end{bmatrix}, \tag{3}$$

which we can use later to find the projected canonical basis $\hat{\mathbf{x}}_{2\mathrm{D}}$, $\hat{\mathbf{y}}_{2\mathrm{D}}$, $\hat{\mathbf{z}}_{2\mathrm{D}}$ on the image plane (see Inglis [5]).

With our frame of reference in place, we are now ready to build mesh models of our impossible figures directly in a 3D environment. For convenience, the coordinate systems defined in this section coincide with the 'Global' and 'Camera View' transformations of the software Blender.



(a) **(b)**

Figure 3: *Coordinate systems as seen from: (a) an arbitrary point of view, (b) the camera point of view.*

Modeling the Necker Cube - A Depth Based Contradiction

The Necker Cube is a 2D drawing depicting the edges of a standard cube, which due to its ambiguity can be interpreted in multiple ways. In particular, one of the 3-dimensional interpretations consists of a series of connecting bars, where the intersecting edges (from the viewer's perspective) appear to be swapped in depth, that is, the edge that should be in the background appears in the foreground instead, thus creating the impression of an impossible object.

We will base our modeling process on this idea and give the viewer a misleading impression of the figure's depth by manipulating local segments of our mesh. Thus, the algorithm to construct the 3D version of the Necker Cube can be broken down as follows, with each corresponding step illustrated in Figure 4:

a) First, create a 'possible' version of the Necker Cube, where each edge corresponds to a 3-dimensional bar, and select one segment of the mesh where the depth will be manipulated.

b) Split the highlighted segment from the main mesh and displace its position $\vec{P}$ towards (or away from) the camera, along the line of the vector $\hat{\mathbf{w}}$ through the operation:

$$\vec{P}_{\text{displaced}} = \vec{P}_{\text{original}} \pm d_{\text{depth}}\, \hat{\mathbf{w}}, \tag{4}$$

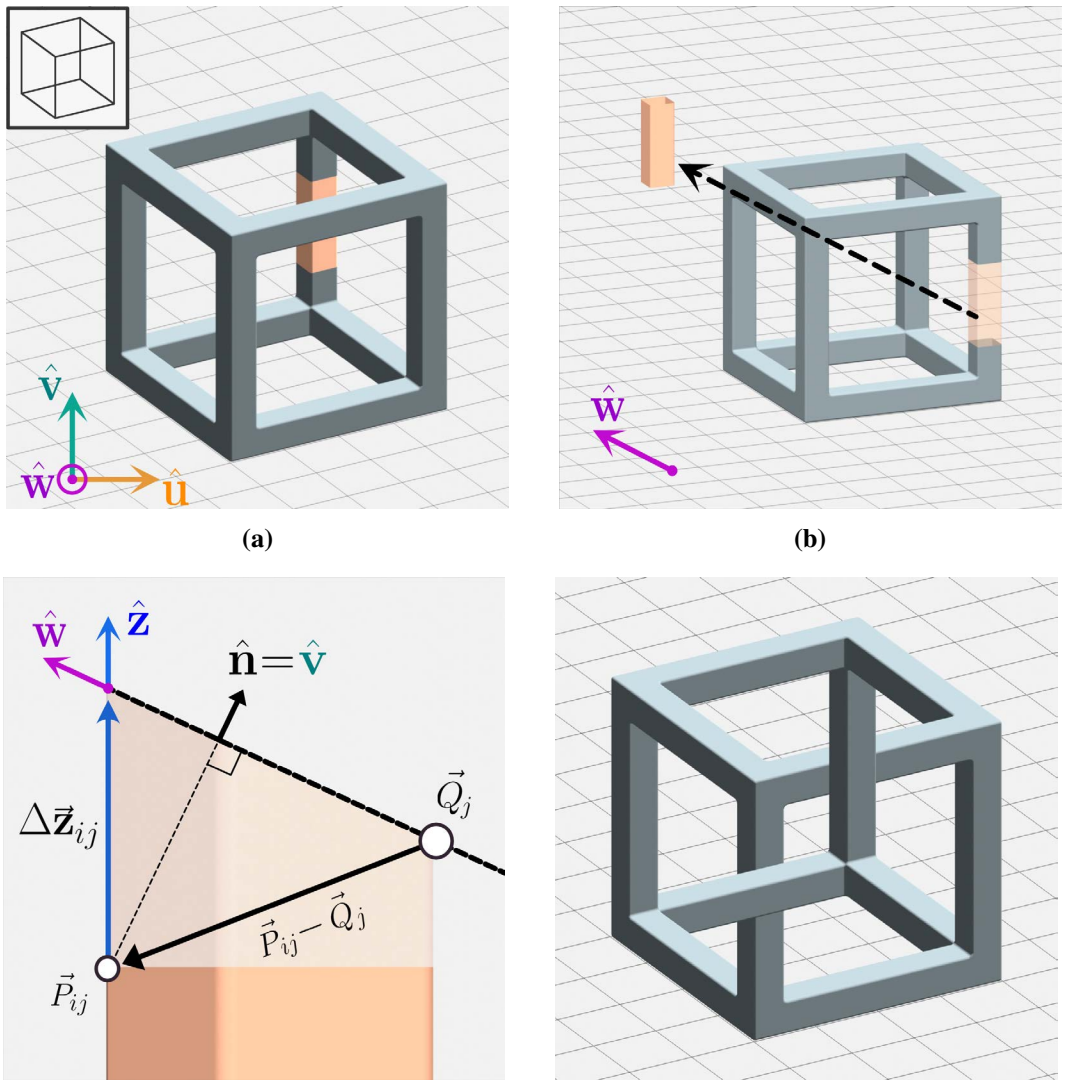
where distance d_{depth} is large enough to bring the segment to the foreground (or background).

c) Finally, align the vertices at the endpoints of the disconnected segment with the camera line of sight, such that the viewer cannot see the discontinuity with the main mesh. Thus, for each endpoint 'j', we displace the corresponding vertices $\vec{P}_{ij}$ along the bar main axis $\hat{\mathbf{z}}$, until they are aligned with a plane of normal $\hat{\mathbf{n}} \perp \hat{\mathbf{w}}$ (such as $\hat{\mathbf{n}} = \hat{\mathbf{v}}$) and convenient origin $\vec{Q}_j$. The displacement vector $\Delta\vec{z}_{ij}$ is then:

$$\Delta\vec{z}_{ij} = -\frac{(\vec{P}_{ij} - \vec{Q}_j) \cdot \hat{\mathbf{n}}}{\hat{\mathbf{z}} \cdot \hat{\mathbf{n}}}\, \hat{\mathbf{z}}. \tag{5}$$

d) Repeat the steps above to include a second paradoxical segment and increase the optical impact of the Necker Cube. For rendering, we recommend using a 'Sun Lamp' (Blender's global directional light) with shadows disabled, in order to keep the shading uniform and hide the discontinuities in the mesh.

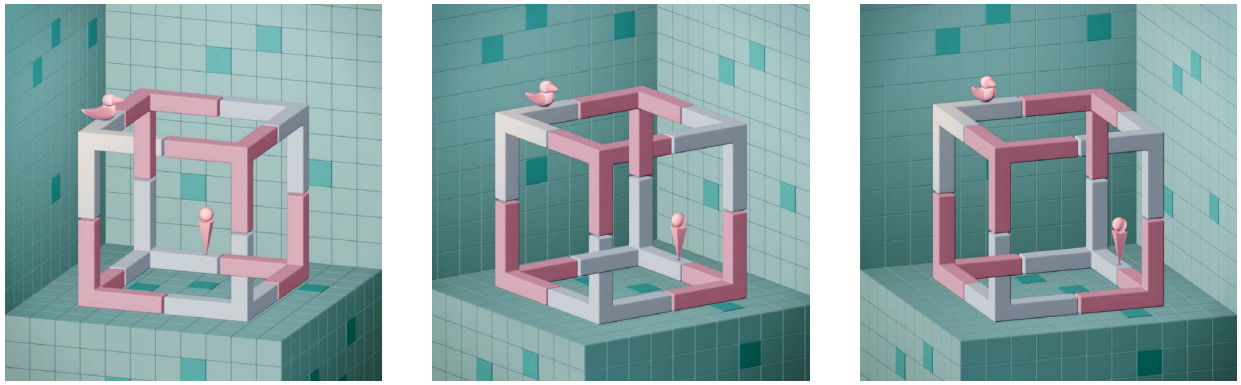
Notice that, because the camera basis vectors $\hat{\mathbf{u}}, \hat{\mathbf{v}}, \hat{\mathbf{w}}$ are defined in function of the angles θ, ϕ (see Equation 2), all the steps above can be parameterized and applied instantly according to the camera orientation. We implement this method in Blender and render an animation of a camera orbiting the Necker Cube. Figure 5 shows three selected frames, along with their corresponding camera orientation.



(a) **(b)**

(c) **(d)**

Figure 4: *Illustration of the steps required to 3D model the Necker Cube. Additional comments to each step can be found in the supplementary material.*



(a) $\phi = 20°$, $\theta = 75°$. **(b)** $\phi = 55°$, $\theta = 75°$. **(c)** $\phi = 75°$, $\theta = 75°$.

Figure 5: *Turn around of Necker Cube. Animation available in supplementary material.*

Modeling the Impossible Bar - An Orientation Based Contradiction

The Impossible (mono)Bar (which is described in the book of B. Ernst [2]) is a particularly deceptive figure due to its simplicity. It consists of a single straight rectangular column, where both end-points are facing towards (or away from) the viewer, and thus creating the impression that the observer is located simultaneously above and below the object.

To model the Impossible Bar in 3D, we will split the mesh into two segments with different orientations, while at the same time convincing the viewer that these still form a continuous straight object. The process is illustrated in Figure 6, and described as follows:

a) We start with a model of a trivial rectangular bar, aligned along the $\hat{\mathbf{z}}$ axis, and split it into its top and bottom segments. The top segment will remain as it is, while the bottom segment will be rotated along a convenient axis.

b) The next step is to find a rotation axis $\hat{\mathbf{k}}$, which must be perpendicular both to the line-of-sight vector and to the bar primary axis (i.e. $\hat{\mathbf{k}} \perp \hat{\mathbf{w}} \perp \hat{\mathbf{z}}$). Therefore, we can easily find $\hat{\mathbf{k}}$ through:

$$\hat{\mathbf{k}} = \frac{\hat{\mathbf{w}} \times \hat{\mathbf{z}}}{\|\hat{\mathbf{w}} \times \hat{\mathbf{z}}\|}, \tag{6}$$

and obtain that for this particular case $\hat{\mathbf{k}} = -\hat{\mathbf{u}}$.

We note that the axis $\hat{\mathbf{k}}$ has the particular property that, upon performing a rotation $R_k(\alpha)$, vertices that were originally aligned along the $\hat{\mathbf{z}}$ axis will remain aligned in the camera view, or in other words:

$$(R_k(\alpha)\,\hat{\mathbf{z}})_{2\mathrm{D}} \parallel \hat{\mathbf{z}}_{2\mathrm{D}}. \tag{7}$$

This means that, even after rotating the bottom segment, the viewer will continue to see the bar aligned along the $\hat{\mathbf{z}}_{2\mathrm{D}}$ axis on the image plane.

At this stage it is also necessary to align the vertices at the middle of the bar with the camera line-of-sight, such that the viewer does not notice the separation between the two segments, following the same method that we used to hide the discontinuity in the Necker Cube (Equation 5).

c) To prevent the viewer from noticing that the bottom segment is rotated due to shading effects (i.e. the interaction of light with the 3D surface), we need to 'freeze' the corresponding face normals prior to the rotation $R_k(\alpha)$. This way, we instruct the render engine to continue shading the top and bottom segments equally, even if they are oriented in different directions.

d) Finally, while the value of α is in principle arbitrary, we have found that the optical illusion looks more consistent if the viewer appears to be equally above and below the object, which is achieved by using a rotation angle of $\alpha = \pi - 2\theta$ (for a bar aligned with the $\hat{\mathbf{z}}$ axis).

We can further increase the sense of spatial contradiction in the viewer if we additionally restrict the camera angles ϕ, θ to a sub-domain where the projected vectors $\hat{\mathbf{x}}_{2\mathrm{D}}$, $\hat{\mathbf{y}}_{2\mathrm{D}}$ become parallel to their rotated cross-counterparts, that is:

$$\begin{aligned} (R_k(\alpha)\,\hat{\mathbf{x}})_{2\mathrm{D}} &\parallel \hat{\mathbf{y}}_{2\mathrm{D}}, \\ (R_k(\alpha)\,\hat{\mathbf{y}})_{2\mathrm{D}} &\parallel \hat{\mathbf{x}}_{2\mathrm{D}}. \end{aligned} \tag{8}$$

For a bar aligned with the $\hat{\mathbf{z}}$ axis, the angles that satisfy this condition are $\phi = 45°$ and $\theta \in (0°, 90°)$. Though these constraints are optional, we think they are worthy of consideration for artistic renderings.

With the algorithm described above it is possible to recreate the Impossible Bar shown in Figure 2(b), and render it from different angles as shown in Figure 7. When animated along the co-latitude angle, the figure in question appears to transition from a configuration that is initially possible ($\theta = 90°$), and move towards configurations that are actually impossible ($\theta < 90°$). An additional variation of the impossible bar (oriented along the $\hat{\mathbf{x}}_{2\mathrm{D}}$ axis) is included in the supplementary material.

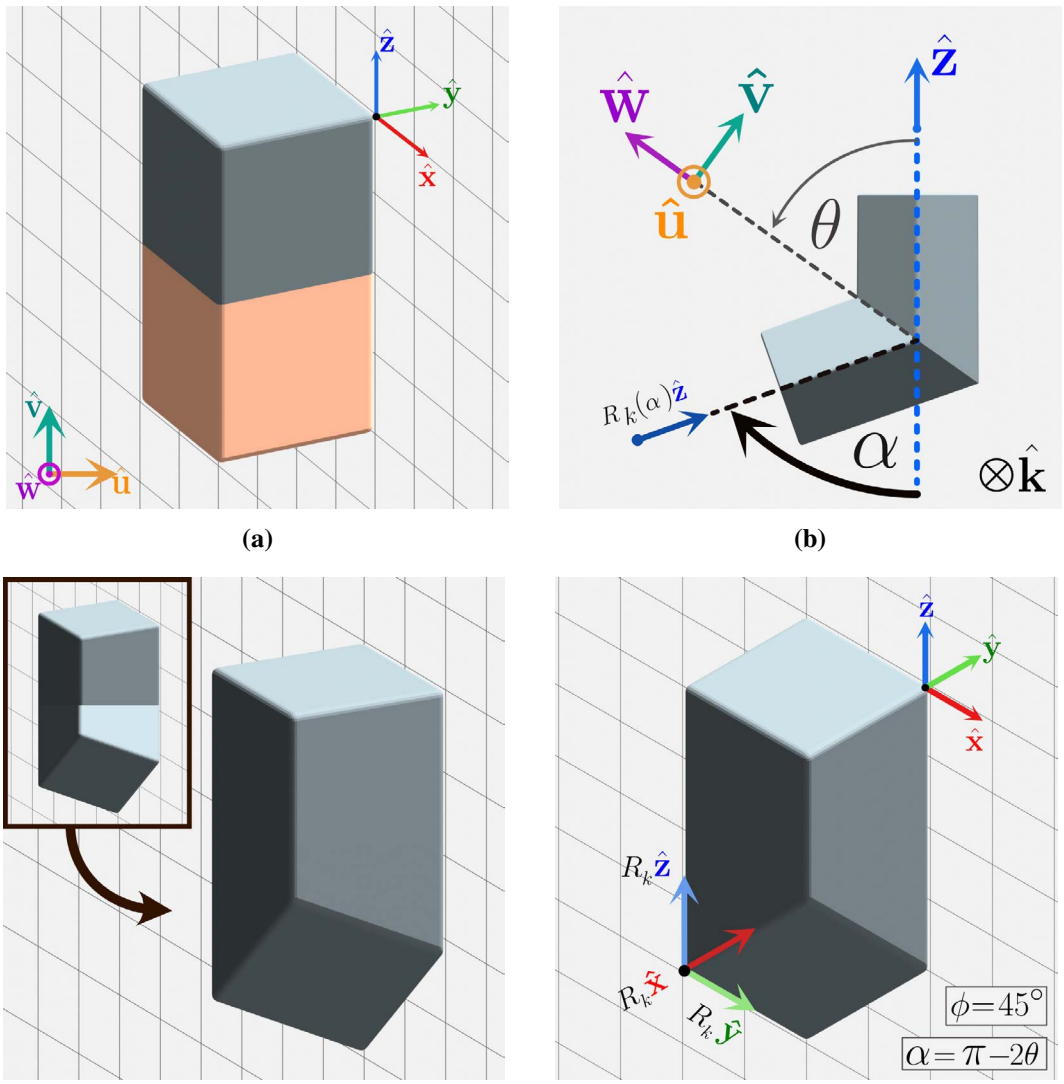
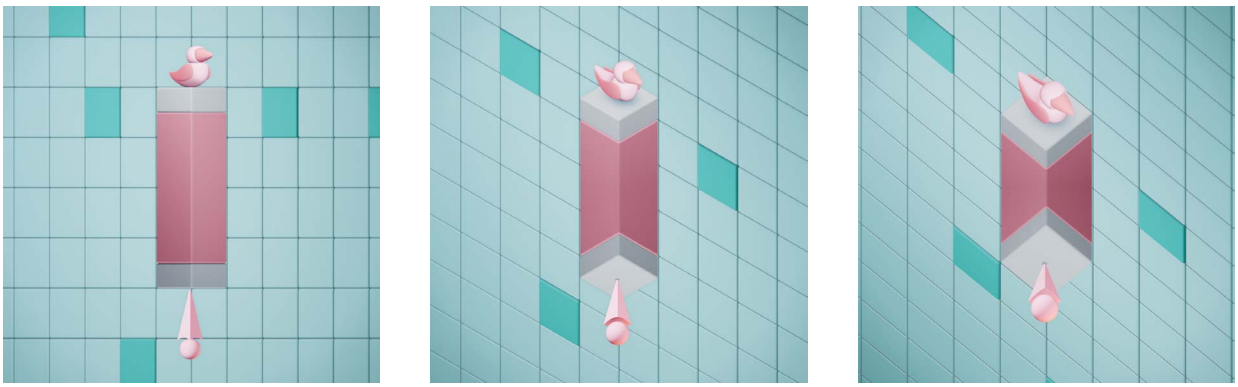


Figure 6: *Illustration of the steps required to 3D model the Impossible (mono)Bar. Additional comments to each step can be found in the supplementary material.*



(a) $\phi = 45°$, $\theta = 90°$. **(b)** $\phi = 45°$, $\theta = 54.736°$. **(c)** $\phi = 45°$, $\theta = 35°$.

Figure 7: *Impossible Bar seen from different orientations. Animation available in supplementary material.*

Interactive Application

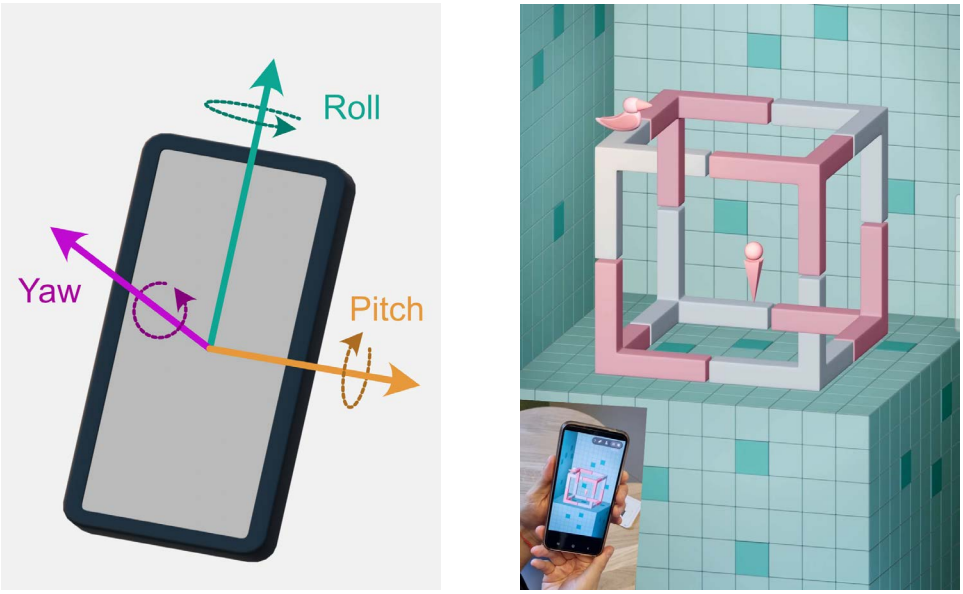
Artworks of impossible figures strongly rely on how the audience perceives the spatial contradiction associated to these objects, and while the camera animations presented in this paper add one extra layer of depth by showing the viewer what these figures would look like from multiple angles (as seen in Figure 5 and Figure 7), we believe that this experience can be taken even further.

We asked ourselves: What if, instead of playing a predefined animation, we updated the figure according to the physical orientation of the screen, relative to the position of the viewer? The result would be an interactive artwork where the viewer can perceive the paradoxical figure as an extension of the physical 3D space! Then with this goal in mind, we implemented an Android App using Processing [3], to showcase an impossible figure that reacts to the rotation of the user's smartphone.

Our solution consists of using the accelerometer sensor of the phone to infer the physical orientation of the screen. To keep our experiment simple, we choose to limit the interaction of the viewer exclusively to rotations around the 'Roll' axis of the device (see Figure 8(a)), and map the 'Roll' angle linearly to our virtual camera azimuth through $\phi = \text{Roll} + 45°$. Then, we also assume that the viewer is holding the phone at a slightly slanted angle (Pitch $\approx 15°$), and consequently lock the value of the virtual colatitude to $\theta = 90° - \text{Pitch} = 75°$ within the App. Finally, our implementation ignores rotations around the 'Yaw' axis, and we instruct the user (verbally) to hold their device in portrait mode.

To conclude, rather than re-creating the modeling algorithm in Processing, we opted instead for loading a sequence of 60 pre-rendered frames from Blender (of the Necker Cube in this example) covering the angle domain $\phi \in [15°, 75°]$ and $\theta = 75°$, and display the frame that coincides with the rotation of the smartphone (see Figure 8(b)). Though this implementation limits the quantity of viewing angles available to the user, it guarantees that the image displayed will be of high render quality, while also reducing the amount of calculations required within the App.

As a result, the viewer will be able to see the Necker Cube from different (azimuthal) angles as they rotate their phone, giving them the impression that the figure itself was embedded inside the screen.



(a) **(b)**

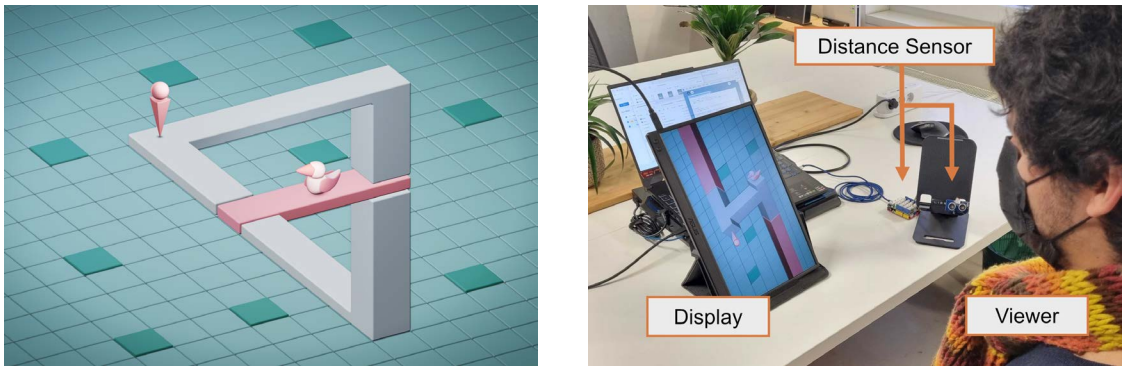
Figure 8: *(a) Schematic of the smartphone rotation axes. (b) Demonstration of the interactive the app available in supplementary material.*

Summary and Further Work

In this paper, we explored a new 3D modeling technique that adds a new dimension to the experience of impossible figures. First, we described a parametric algorithm to build mesh models of the Necker Cube and the Impossible Bar that can be directly rendered from an arbitrary camera angle; we showed that this algorithm can be used to create artistic versions of impossible figures in combination with the standard 3D modeling techniques (particularly within Blender); and finally, we implemented an interactive Android application in Processing, where the viewer can turn around an impossible figure by physically rotating their smartphone.

The modeling techniques presented here can be further extended to parameterize other kinds of impossible objects, such as the Penrose Triangle (see Figure 9(a), additional examples are available on the website of M. Gárate [4]); and the interactive implementation can also be adapted to create immersive experiences on larger scale displays, for example by using an arrangement of distance sensors to detect the position of the viewer in a room where the paradoxical figure is displayed (see a prototype for this concept in Figure 9(b)).

As an art form, impossible figures present the audience with an unique kind of challenge to their understanding of 3-dimensional space; they create a sense of spatial contradiction that captures the viewer's curiosity and leads them to keep looking, if only for a few extra seconds, at these illustrations of paradoxical worlds. We hope that this work can expand the possibilities of collaboration between artists, mathematicians, and engineers; and serve as a guideline for future endeavors in the creation of interactive impossible figures.



(a) **(b)**

Figure 9: *(a) Art variation of the Penrose Triangle. (b) Prototype of an interactive paradox with an Arduino controller and a distance sensor.*

References

[1] Blender Foundation. *Blender, a 3D modelling and rendering package*. 2002. https://www.blender.org/

[2] B. Ernst. *Adventures with Impossible Figures*. Tarquin Publications, 1986.

[3] B. Fry and C. Reas. *Processing*. 2001. https://processing.org/

[4] M. Gárate. *Matías Gárate*. 2025. https://www.matiasgarate.com/

[5] T. Inglis. "Constructing Drawings of Impossible Figures with Axonometric Blocks and Pseudo-3D Manipulations." *Bridges Conference Proceedings*, Seoul, Korea, 14–19 August, 2014, pp. 159-166. https://archive.bridgesmathart.org/2014/bridges2014-159.html

[6] L. S. Penrose and R. Penrose. "Impossible Objects: A special type of visual illusion." *British Journal of Psychology*, vol. 49, no. 1, 1958, pp. 31–33.