Exploring Shader Based Non Planar Projections in Unity

A Comparison of Object Based and Object Independent Projections

Javier Rodríguez Lavandeira
Henry Albert
MSc / Virtual and Augmented Reality
MSc Computer Games Programming
Module: Mathematics for Games and V/AR
Goldsmiths, University of London

Date: January 26, 2026

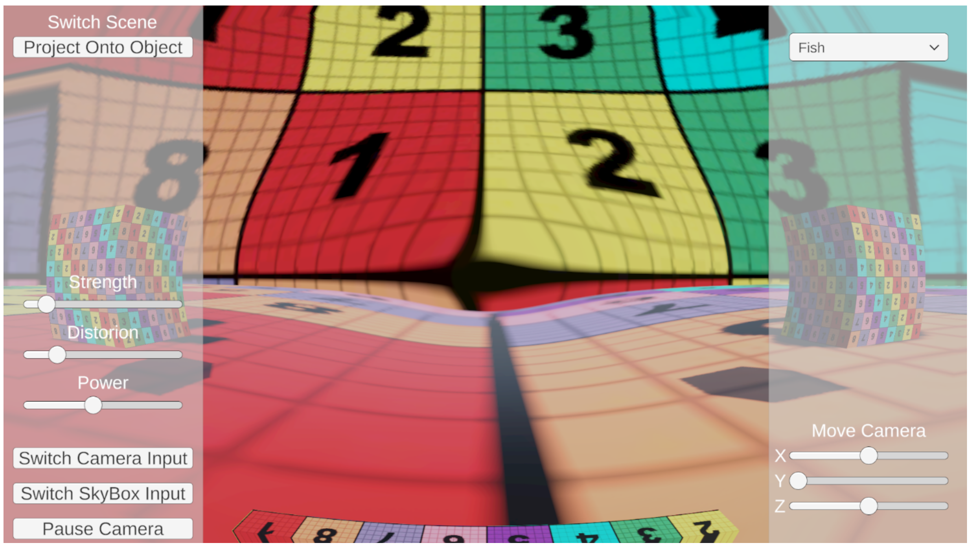


Figure 1: Fisheye object independent projection

- Video: https://youtu.be/7DkoOHFSxGw
- GitHub: https://github.com/HenryAlbert55/ExploringShaderBasedNonPlanarProjections

Abstract

This project investigates the implementation of non planar projection techniques in Unity, focusing on spherical, conical, and cylindrical projections. Each projection model was implemented using two different approaches: projection applied onto a 3D object (Object based) and projection performed without an intermediary object (Object independent). This distinction enables a comparative evaluation of visual distortion and spatial continuity.

Contents

1 Introduction

Each projection model was implemented using two distinct approaches. The first approach applies the projection onto a three dimensional geometric surface, referred to as the object based method, where the rendered image is mapped directly onto a corresponding 3D object (e.g., a sphere or cone). The second approach, termed object independent, performs the projection mathematically at the rendering stage without relying on an intermediary 3D object, instead modifying the projection calculations directly.

By implementing both approaches for each projection type, the project enables a comparative analysis of their visual and spatial characteristics. The evaluation focuses on differences in visual distortion, spatial continuity, and overall image coherence across the projection methods and implementation strategies.

1.1 Background

Ideas of non planar projection date back centuries in art and cartography, The first ever non planar projection existed long before computers, dating back to the Renaissance, where artists explored curvilinear projections [5] like the fisheye, which can be seen in the painting drawn by Jan van Eyck (figure 2).

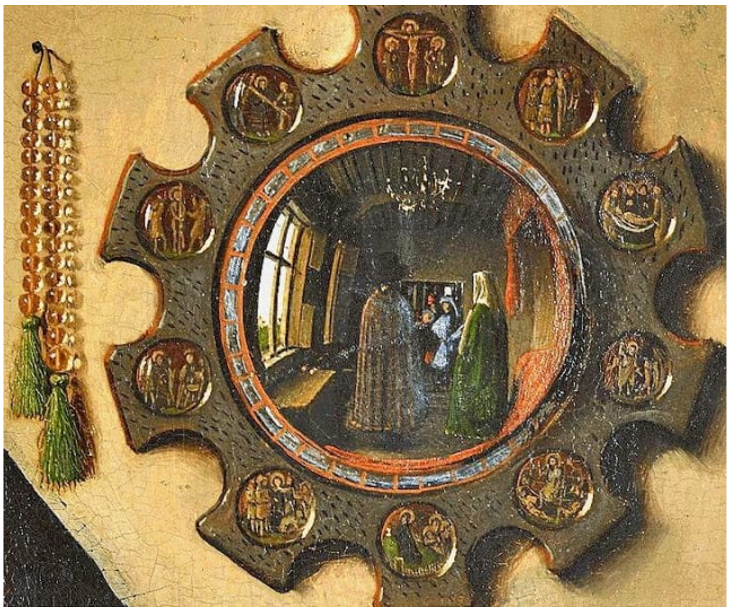


Figure 2: The Arnolfini Portrait

After this, the first uses of spherical projections were used in the 19th century in carthography [1], allowing the humanity to represent the planet Earth (A 3D "sphere") in a plane. An example of this is Peirce quincuncial projection in 1877 (figure 3)

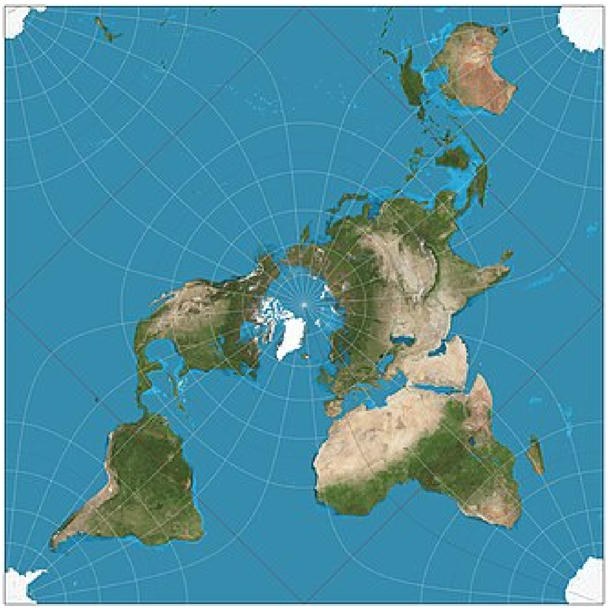


Figure 3: Peirce quincuncial projection

So the idea of projecting onto curved surfaces or using curved projection models long predates computer graphics.

The earliest documented example of non planar projection onto a non flat surface in an experiential context was in the 1969, at the opening of DisneyLand's Haunted Mansion ride (figure 4), where film was projected onto sculpted surfaces. This is not a computer graphics rendering technique yet, but it is significant as the first intentional use of projection on 3D shapes in a real environment.

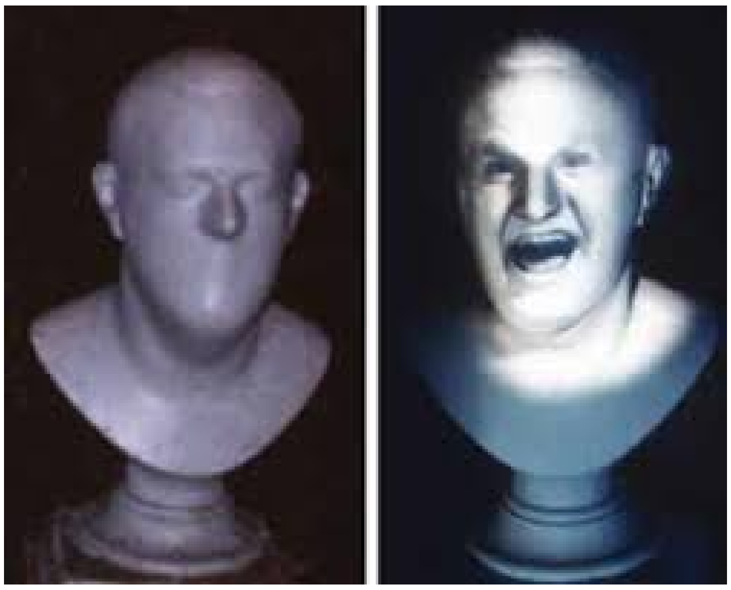


Figure 4: DisneyLand's Haunted Mansion

This techniques started to appear in computing graphics research in the late 1990-2000s, and have been developing and being used until now.

1.1.1 Pinhole Camera

The pinhole camera represents one of the earliest known models for understanding image formation and projection, forming the conceptual foundation of modern optical systems and computer graphics. Its underlying principle, that light travels in straight lines and can form an inverted image when passing through a small aperture, has been observed and documented for over two millennia [8].

During the Renaissance, the pinhole camera gained prominence in Europe as both a scientific instrument and an artistic aid. Figures such as Leonardo da Vinci expanded on the camera obscura's principles, explicitly comparing the device to the human eye and using it to study perspective, proportion, and light. Renaissance artists employed camera obscura devices to assist with accurate spatial projection, reinforcing the link between optical projection and linear perspective in visual representation [2].

In the 20th and 21st centuries, the pinhole camera model became fundamental to computer vision and computer graphics. In these fields, it serves as an idealized camera model for simulating image formation, defining projection matrices, and mapping three dimensional coordinates to two dimensional image planes. Modern rendering pipelines, including those used in real time engines such as Unity, are built upon perspective projection models that directly descend from pinhole camera geometry.

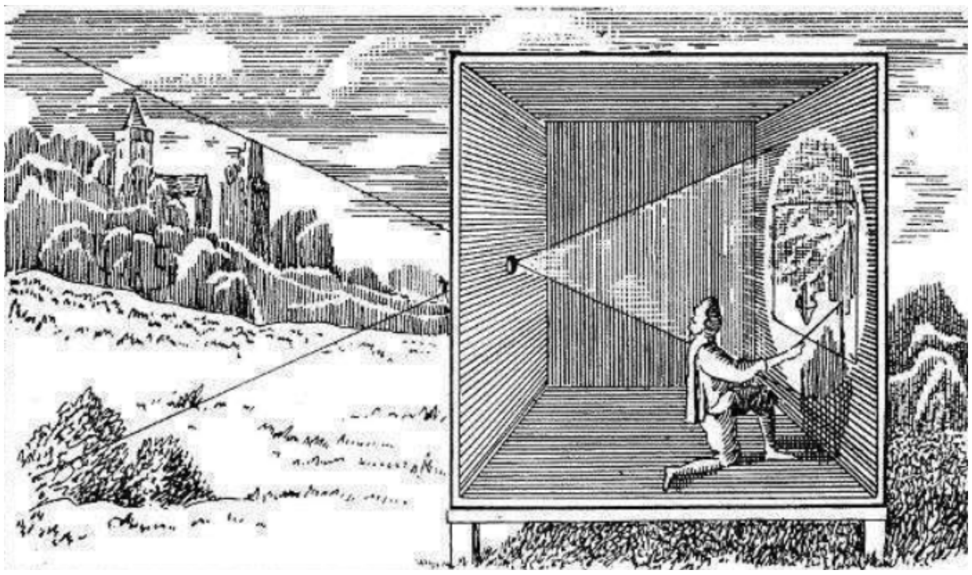


Figure 5: Camera Obscura

1.2 Perspective Projection

Real time rendering systems typically rely on planar projection models, such as perspective projection, to display three dimensional scenes on two dimensional screens by projecting each point along a line that passes through a single centre of projection. These techniques are derived from Renaissance knowledge of perspective that can be observed in many famous artworks, such as Holbein's The Ambassadors (figure 6a) and Raphael's The School of Athens (figure 6b).

(a) The Ambassadors

(b) The School of Athens

Figure 6: Renaissance Perspective Art

The transformation is defined by a perspective projection matrix, in which the projected coordinates are scaled by the inverse of their depth value, producing depth dependent foreshortening and convergence toward vanishing points. In real time graphics systems such as Unity, this projection is implemented in the rendering pipeline by transforming vertices from view space to clip space and performing a homogeneous divide, resulting in normalized device coordinates suitable for rasterization [4].

Inversive geometry, particularly the concept of inversion in a sphere, offers a mathematical framework that shares conceptual similarities with non planar projections. Inversion swaps near and far regions of space relative to an invariant sphere, producing highly non linear mappings that preserve angles locally (conformal mapping) [6]. Although our project focuses on direct spherical and conical projections rather than inversion, the underlying idea of transforming the geometry of space to achieve a different perspective aligns with our shader based projection models. In particular, the "spherical perspective" described in inversion geometry is comparable to the spherical cubemap sampling implemented in our object independent shaders, where the image is mapped onto a spherical field of view and depth information is effectively compressed.

2 Method

2.1 Our Work - Description of the project

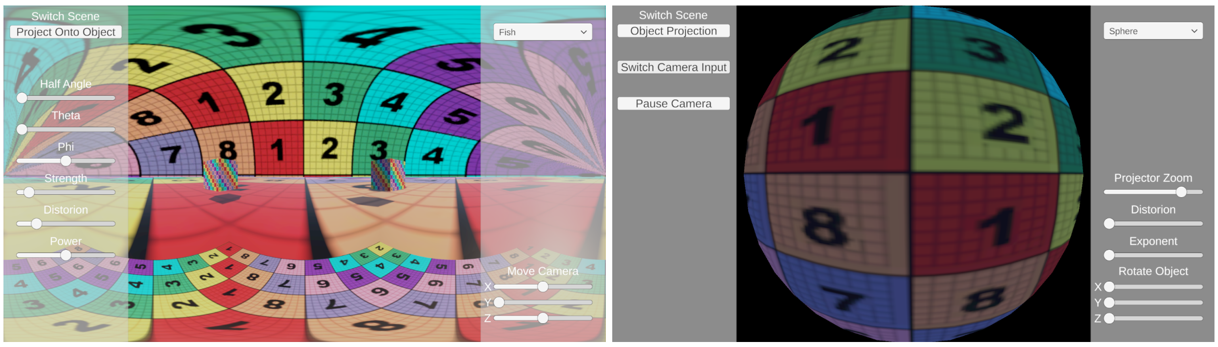
We used the Unity game engine to create different perspective projections by writing custom shaders [7]. We focused on two types of non planar projections, making one scene for each.

The first approach, referred to as object independent projection, generates non planar projections directly in screen space using mathematical mapping rather than projecting onto an intermediary 3D object. In these shaders, each screen pixel's UV coordinates are converted into a direction vector through spherical or conical mapping functions. The direction vector is then used to sample a cubemap texture representing the captured environment, producing an image that appears projected onto a curved surface. This method treats the projection as a purely mathematical transformation from 2D UV coordinates to 3D direction space, resulting in a projection that is independent of any target geometry and entirely defined by the projection model itself.

The second approach, referred to as object based projection, projects a texture onto non planar 3D geometry using the object's surface as an active component of the projection process. In this case, the image is mapped from a virtual projector located at a defined world position, with the projector's view projection matrix determining how each point on the geometry is transformed into projector space. Pixels that fall outside the projector's frustum or face away from the projector are discarded, ensuring the projection only appears on visible surface regions. Additionally, the shader applies geometry aware distortion based on the angle between the surface normal and the projector direction, producing a curved, fisheye like deformation that increases with grazing angles. This method therefore produces a projection that is inherently dependent on the shape and orientation of the target object.

2.1.1 UI

As we had two different scenes, we had to make two different UIs, being this quite similar but with different parameters for each scene and/or each type of projection.



(a) Object independent projection (b) Object based projection

Figure 7: UI examples

For the object independent projection (figure 7a):

- Half Angle (used on the cone projection)
  Changes how large the cone angle is, the default is 45° so the full angle is 90°. When the half angle approaches 90° the image gets extremely stretched, and when over 90° it is inverted.

- Theta (used on the sphere, cone and waterdrop projection)
  Changes the longitude, the multiplier that by default turns the 0 to 1 uv coordinates into -180° to +180° for 360° camera sides. When increased over 2 the projection begin to repeat itself along the horizontal axis as the longitude is larger than 360°. When decreased below 0 the result is inverted. When between -2 and 2 it changes the longitude to be less than a full 360°.

- Phi (used on the sphere, cone and waterdrop projection)
  Changes the latitude, the multiplier that by default turns the 0 to 1 uv coordinates into -90° to +90° for 180° camera top/bottom. When increased the projection repeats itself along the vertical axis as the longitude is larger than 360°. When decreased below 0, the projected result gets inverted. When between -1 and 1 changes the latitude to be less than a full 180°.

- Distortion, strength and power (used on the fish and waterdrop projection)
  These parameters control a radial distortion applied to the texture coordinates. The distortion is defined as a non linear transformation that increases with the distance from the centre of the image. The strength and power parameters determine the intensity and spatial distribution of this distortion. The `_Distortion` parameter

provides a baseline scaling factor, while `_Strength` controls the magnitude of the distortion as a function of distance. The `_Power` parameter adjusts how rapidly the distortion increases, allowing the effect to be concentrated near the edges or distributed more evenly across the image.

```
uv *= _Distortion + _Strength * pow(r, _Power);
```

- Move Camera (all)
  Allows the user to move the camera.

For the object based projection (figure 7a):

- Projector Zoom
  Allows the move of the projector on the Z axis
- Distortion
  Controls how strong the geometry based warp is by scaling the uv radially outward. The larger the value the stronger the stretching.

  ```
  centered += centered * geoWeight * _GeoDistortion;
  ```

- Exponent
  Controls the fallof curved based on the angle, it shapes how fast the increase happens. When it is 1 it scales linearly. When its greater than 1 the distortion on the edges suddenly ramps up. When its less than 1 the distortion appears earlier.

  ```
  float geoWeight = pow(1.0 - facing, _GeoExponent);
  ```

- Rotate Object
  Allows the rotation of the object on which the camera view is projected.

2.2 Decisions made

During the development of the project, several key decisions were made to guide the implementation and evaluation of non planar projections. Initially, the work began with a simple fisheye lens distortion, which provided a baseline for understanding non linear projection effects in real time rendering. We then chose to implement the different projection models primarily through Unity shaders, allowing for direct manipulation of vertex and fragment stages and enabling real time performance and interactivity. To illustrate and compare the resulting projections, we used UV mapped textures that clearly demonstrate how each projection distorts and maps the image onto the target surface. In order to increase the practical applicability of the system, we later added webcam input as an option, enabling live video to be projected in place of the UV textures in the object based projection approach. Finally, although our initial implementation of conical projection did not meet our original expectations, the output produced an unexpected yet visually compelling effect. Rather than discarding this result, we retained it as an additional projection variant, naming it the "waterdrop" (figure 8) projection to reflect its distinctive appearance.

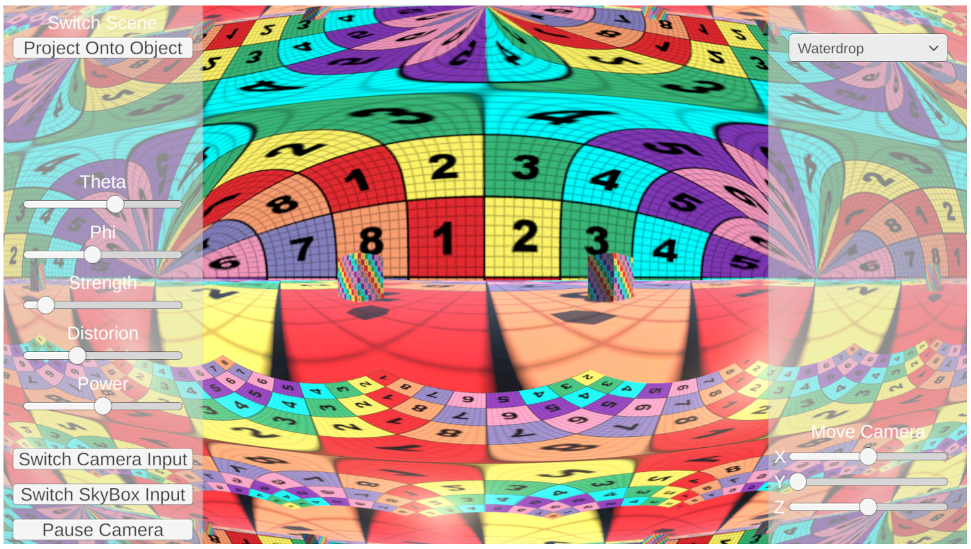


Figure 8: Waterdrop Projection

2.3 Goals

The primary objective of this project was to implement object based projection, with an initial focus on spherical surfaces. During development, however, the scope expanded when the implementation of a shader based projection system revealed the potential for object independent projection. As a result, the project evolved to include both object based and object independent projection techniques, enabling a broader comparison of the two approaches.

In addition to spherical projection, a stretch goal was established to explore alternative non planar projection models, specifically conical and cylindrical projections. These additional projection types were intended to test the flexibility of the implementation and to assess how different projection geometries affect visual distortion and spatial continuity. Another stretch goal was to combine these projections with the dolly zoom [3] effect devised for Alfred Hitchcock's film Vertigo, as we believed it could have an interesting result.

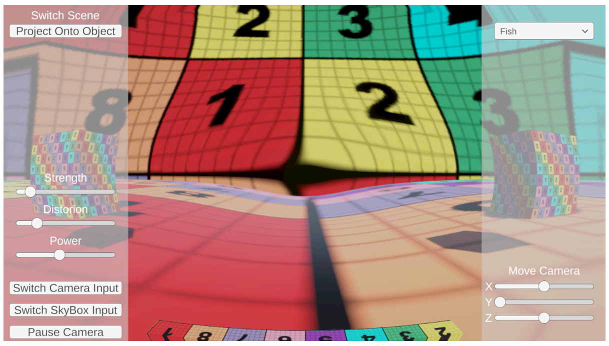
A further aim was to develop the system as an interactive tool, providing users with extensive control over projection parameters. Adjustable variables such as distortion, strength, projection angles, geometric exponent, and camera location were exposed to the user. The tool was designed to support experimentation and facilitate deeper insight into the behaviour of non planar projections in real time rendering environments.

3 Team

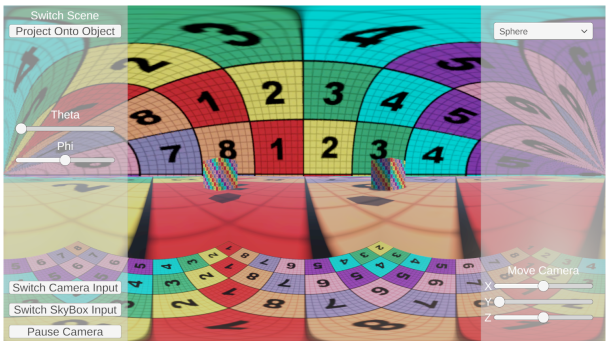
Both participants collaborated on all aspects of the project, including research, scene creation, shader development, and testing. Unity's version control system was used to maintain project integrity, track changes, and allow simultaneous editing of scenes and assets.

4 Results

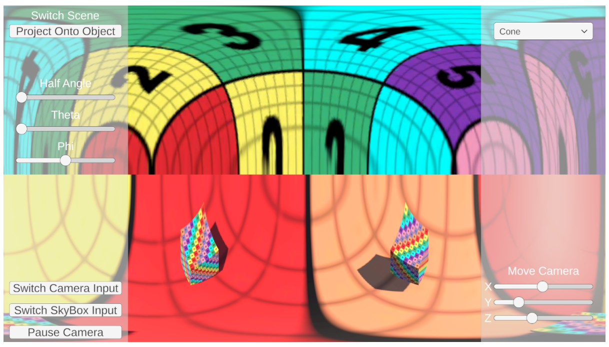
This section presents the visual results obtained from the implemented non-planar projection techniques.



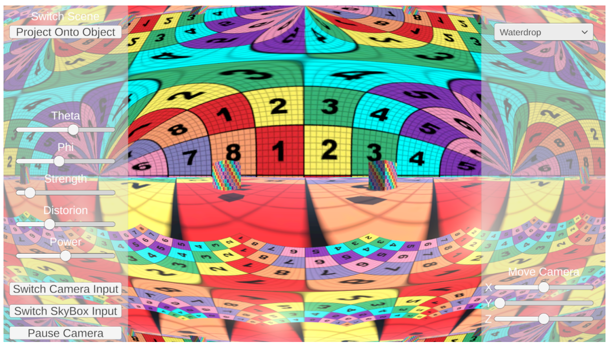
(a) Fish eye



(b) Sphere



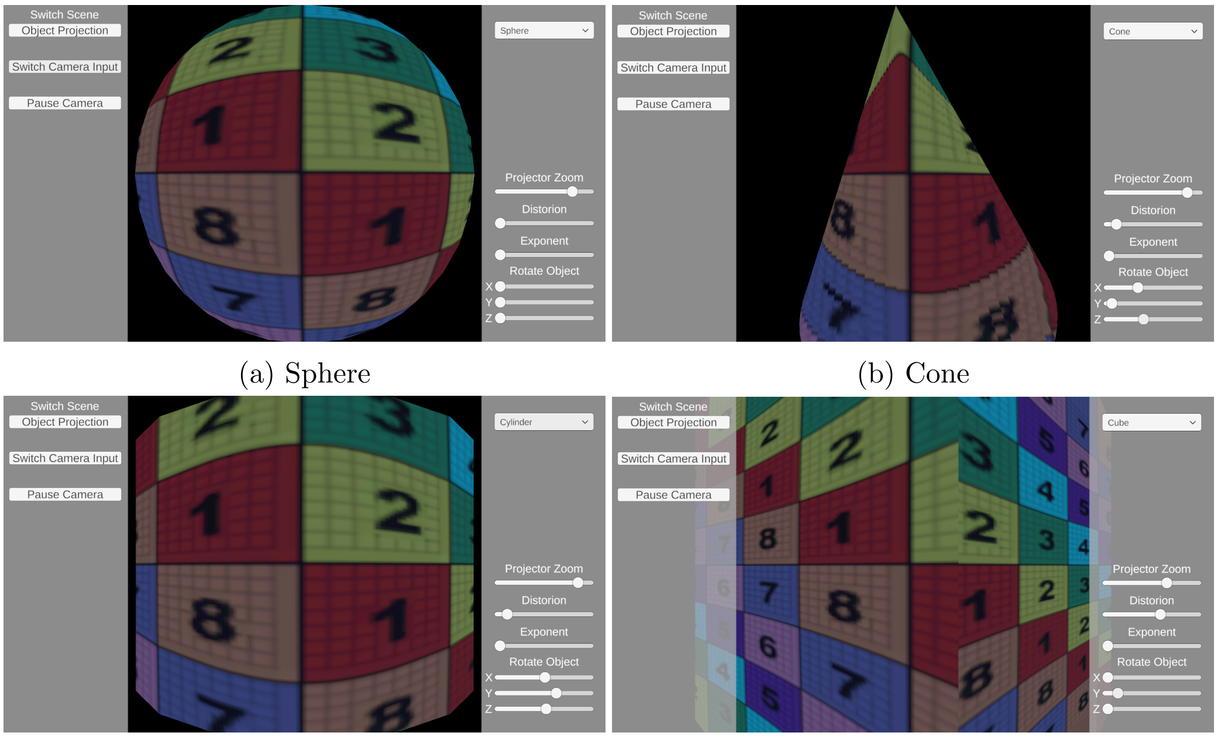
(c) Cone



(d) Waterdrop

Figure 9: Object Independent Projection Results

Figure 9 shows the object-independent projection results that were rendered onto a planar surface. The spherical projection produces a uniform radial distortion, preserving central features while increasingly compressing geometry toward the edges. The conical projection introduces asymmetric distortion, with stretching concentrated along the cone axis. The top and the bottom of the cone appear independent as when viewed from a cone they are not be visible to the other. The fisheye projection exhibits strong non linear distortion near the image boundaries, significantly expanding the field of view at the cost of spatial compression. The waterdrop combines a slightly altered version of the spherical and fish eye projections creating a uniform radial distortion with strong non linear distortion near the image boundaries. We interpret this as similar to what we imagine a projection from inside a water drop falling into a pool of water would be.



(a) Sphere (b) Cone

(c) Cylinder (d) Cube

Figure 10: Object Based Projection Results

The figure 10 presents the object based projection results, where the camera view is mapped directly onto non planar geometry. In these cases, the projection conforms to the surface of the object, resulting in a continuous image across the geometry.

5 Conclusion

The application successfully meets most the goals of the project (section 2.3). We managed to implement object based projection as well as object independent For the object independent we implemented spherical and conical projections as well as fisheye projection to explore the limits of the non planar projections. As a result of mixing and adjusting the spherical and fisheye projections we achieved an interesting result, although it was not the conical projection that we were expecting, obtaining a unique projection which reminded us of a waterdrop. For the object based we implemented the projections over 3D shapes such as sphere, cone, cube and cylinder.

The implementation of a interactive UI was also done, completing so far the second goal that we had. We added different sliders for changing the parameters of the projections, making the application a very personalisable tool. For ensure the correct performance of the project, we used some UV mapping textures that demonstrated how the projection was distorting the camera view. In addition to the goals mentioned, we added the option of using the laptop camera to test the projections from the object based scene and to widen the applications of the project. From the goals mentioned we couldn't fulfil all of them as we didn't add the dolly zoom effect [3]. Additionally, if we were to continue this project, exploring inversive projections could be a future extension to further expand the range of non planar projection effects.

References

[1] Projection Mapping Central. The illustrated history of projection mapping. https://projection-mapping.org/the-history-of-projection-mapping/, 2026.

[2] Perspective Research Centre. Origins – perspective research centre. https://perspectiveresearchcentre.com/history-2/, 2025.

[3] Wikipedia Contributors. Dolly zoom. https://en.wikipedia.org/wiki/Dolly_zoom, 2026.

[4] Karen Foley. 3d math primer for graphics and game development — more on matrices — perspective projection. https://gamemath.com/book/matrixmore.html#perspective_projection, 2011.

[5] Grokipedia. Curvilinear perspective. https://grokipedia.com/page/Curvilinear_perspective, 2026.

[6] Jan Koenderink. Hold infinity in the palm of your hand. https://gestaltrevision.be/wp-content/uploads/clootcrans_press/2024_HoldTheWorld.pdf, 2023.

[7] Unity Technologies. Vertex and fragment shader examples. https://docs.unity3d.com/560/Documentation/Manual/SL-VertexFragmentShaderExamples.html, 2019.

[8] Jingyi Yu, Leonard McMillan, and Peter Sturm. Multi-perspective modelling, rendering and imaging. https://scispace.com/pdf/multiperspective-modeling-rendering-and-imaging-1tudvh4d7y.pdf, 2010.