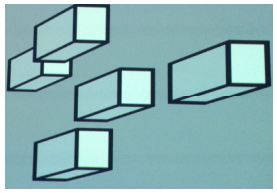
Creating Anamorphic Illusions with Microsoft Kinect

David Spiegel, Nathan Fellom, Nicholas Kryzer, and Dr. Daniel Stevenson

University of Wisconsin – Eau Claire | Department of Computer Science

Introduction

A perspective anamorphosis distorts an image such that it requires the viewer to look from a specific vantage point for the image to appear undistorted. Often seen in sidewalk art, this projection can create an illusion of depth on a flat surface. Our goal was to use data from a Microsoft Kinect to change the vantage point of our anamorphic image to match the position of the viewer. With this technique we can produce the illusion of 3 dimensions without requiring 3D glasses.



A distorted image.

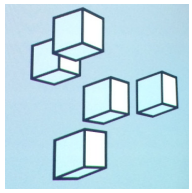
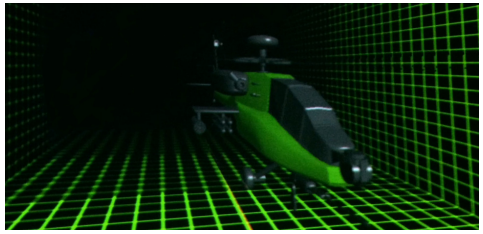


Image from correct vantage point



3D illusion with helicopter

Components

We used the following technology and software in our project:

Technology

- **Microsoft Kinect:** Used to pinpoint the location of the viewer in relation to the screen by tracking skeletal positions.
- **Projector:** Although not needed, we felt it produced a more convincing illusion when projected onto a wall or chalkboard.

Software

- **Unity:** A 3D game engine that allowed us to quickly produce scenes and had available plugins to obtain Kinect compatibility.
- **Math.net Iridium:** We utilized this numerical package for the linear algebra algorithms required in the calibration step.

Approach

Step 1: Calibration

The viewer's position we receive is relative to the Kinect and does not take into account the position of the projector/TV screen. With this in mind, we needed to map Kinect points to where they should appear in relation to the screen. We attempted two methods:

Manually: Setting offset and scaling values with trial and error worked but was tedious to do.

Programmatically: We first obtained a list of Kinect and world point pairs:

$$(x_{k1}, y_{k1}, z_{k1}) \Rightarrow (x_{w1}, y_{w1})$$
$$(x_{k2}, y_{k2}, z_{k2}) \Rightarrow (x_{w2}, y_{w2})$$
$$\dots$$

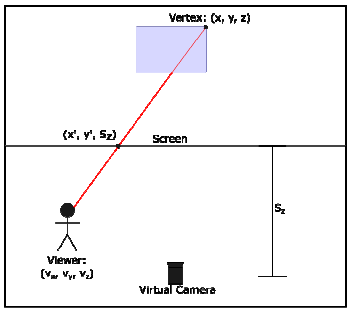
Next we constructed multiple linear equations with coefficients of a transformation matrix.

$$x_{w1} = r_{11}x_{k1} + r_{12}y_{k1} + r_{13}z_{k1} + t_1$$
$$y_{p1} = r_{21}x_{k1} + r_{22}y_{k1} + r_{23}z_{k1} + t_2$$
$$\dots$$

We then wrote these equations in matrix form and estimate the coefficients. This is done by using QR decomposition to minimize the sum of the squared differences between model and data values. We now have a transformation matrix that maps Kinect points to world points.

Step 2: Anamorphic Projection Matrix

- We equate lines from the viewer's world position to each of the vertices that compose a scene.
- The intersection point of a line with the screen gives us the new x, y coordinate for the vertex.
- However, the z-coordinate is not set to be the z-coordinate of the screen. Instead, a value is used where the z ordering of the original model is retained. We do not want to project the entire scene onto a single plane because z-fighting can then occur.



An example scene where one vertex of a box is projected onto the screen

Result:

- With these rules, we calculated a new projection matrix and passed it directly to Unity's camera class.
- With this approach, we are able to quickly and efficiently distort an entire scene based on the player's position. This was superior to our initial implementation of moving each vertex individually on the CPU.

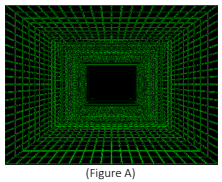
$$P = Ortho * \begin{bmatrix} v_z - S_z & 0 & -v_x & S_z * v_x \\ 0 & v_z - S_z & -v_y & S_z * v_y \\ 0 & 0 & v_z - f - n & f * n \\ 0 & 0 & -1 & v_z \end{bmatrix}$$

The final projection matrix

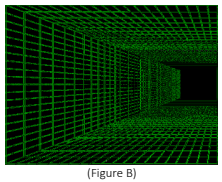
Step 3: Scene Creation

We used certain scene elements to help sell the illusion.

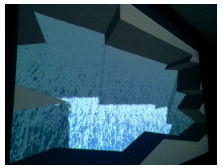
- A tunnel that frames the screen (Figure A). Standing in front of an edge of the tunnel hides it from view (Figure B).
- A background of fog that obscures distant objects. This makes distant objects appear farther away.
- Shadows for objects that are upright.

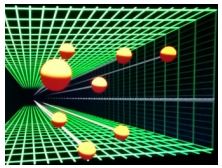


(Figure A)
A tunnel when viewed straight on



(Figure B)
Tunnel viewed from side with edge obscured





Examples of different 3d scenes

Conclusion

We found that the 3D illusion is diminished by the binocular vision of our eyes, which notices that the image is projected onto a flat screen. This means that videos of this effect, which are shot through a monocular lens, look more realistic than they do in real life. In the future, we would like to research different ways to combat this.

Additionally, the viewing angle of the Microsoft Kinect is not ideal. Therefore, we would like to explore different methods of reliably to accurately track the viewer's position.

We thank the Office of Research and Sponsored Programs for supporting this research, and Learning & Technology Services for printing this poster.