Exámen 2
Visión por Computador

Kevin D. Ortega
Daniel Zapata
Daniela Buitrago
Universidad Tecnológica de Pereira
Maestría en Ingeniería Eléctrica
Email: kevin.ortega@utp.edu.co, daniel.zapata1@utp.edu.co, d.buitrago1@utp.edu.co

***Abstract*—Este informe detalla la conceptualización, diseño e implementación del exámen dos de visión por computador, centrado en la geometría proyectiva en el espacio tridimensional P^3. A través de rigurosos métodos matemáticos.**

El objetivo principal consiste en la elaboración del juego Block Out 2 usando Python y algunas librerías como matplotlib, numpy y tkinter. El juego consiste en manipular figuras tridimensionales que caen en un cuarto proyectado en la pantalla. El jugador debe rotar y trasladar las figuras para encajarlas en el fondo del cuarto, evitando que se acumulen y lleguen al techo. Para implementar la parte proyectiva del juego, se deben seguir los siguientes pasos:

- **Definir un conjunto de al menos 3 figuras básicas en 3D, incluyendo al menos una cuádrica.**
- **Proyectar ese cuarto en una imagen en pantalla (como el tablero donde se juega block out).**
- **Escoger una de las figuras básicas y ubicarla en el cuarto proyectado.**
- **Con algunas letras del teclado controlar rotación y traslación de dicho objeto.**

I. Introducción

Este artículo se centra en la implementación del juego *Block Out 2* como un caso de estudio que combina la geometría proyectiva con aplicaciones prácticas en el ámbito de los juegos interactivos. Para lograr este objetivo, se han seguido una serie de pasos clave, incluyendo la definición de figuras tridimensionales en el espacio P^3, la proyección de estas figuras en 2D y su manipulación por parte del jugador.

II. Geometría Proyectiva y Transformaciones

La geometría proyectiva se basa en el concepto de proyección, que es una transformación que mapea puntos de un espacio tridimensional P^3 a un espacio proyectivo bidimensional P^2 [1]–[3]. Esta transformación es fundamental para representar objetos 3D en una pantalla 2D, como se requiere en el juego *Block Out 2*.

La proyección perspectiva se puede expresar matemáticamente como una transformación lineal utilizando una matriz de proyección. Esta matriz se utiliza para llevar a cabo la proyección desde el espacio 3D al espacio 2D, lo que implica la conversión de coordenadas homogéneas tridimensionales a coordenadas bidimensionales. La matriz cuádrica que se expanderá a lo largo de este informe es necesaria para la definición de las figuras [4].

III. Definición de Figuras y Transformaciones Geométricas

En la implementación del juego *Block Out 2*, se han definido figuras tridimensionales, incluyendo cuádricas, como elementos jugables. Estas figuras se representan mediante ecuaciones matemáticas que describen sus formas geométricas en el espacio P^3. Para lograr la interacción con el jugador, se aplican transformaciones geométricas, como rotaciones y traslaciones, a estas figuras.

Las transformaciones geométricas se expresan como matrices de transformación que actúan sobre las coordenadas homogéneas de las figuras. Estas matrices se aplican para modificar la posición y orientación de las figuras en el espacio proyectivo, lo que permite al jugador interactuar con ellas de manera dinámica [5].

La base fundamental para definir las figuras que se implementarán en el proyecto se encuentra en el concepto de un cubo tridimensional. A partir de esta figura geométrica primaria, se desencadenará el proceso de definición y construcción de las geometrías necesarias,el proyecto se centra en la cuádrica y otros elementos geométricos afines.

La cuádrica que representa un cubo en geometría proyectiva se describe mediante una matriz cuadrada llamada matriz de forma cuadrática o matriz cuádrica. Para un cubo, la matriz de forma cuadrática se expresa como:

$$Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & -1 \end{bmatrix}$$

Donde:

- Q es la matriz de forma cuadrática del cubo en geometría proyectiva.
- Q_{11}, Q_{22}, y Q_{33} son 1, indicando que la cuádrica se extiende a lo largo de las tres dimensiones principales del espacio 3D.
- Q_{44} es -1, indicando que la cuádrica se extiende en la dirección opuesta al eje Z.

La ecuación general de la cuádrica para el cubo en geometría proyectiva se expresa como:

$$X^T Q X = 0$$

Donde:

- X es un vector de coordenadas homogéneas que representa un punto en el espacio proyectivo. Para el cubo en 3D, X se define como $X = [x, y, z, w]$.
- Q es la matriz de forma cuadrática del cubo que definimos anteriormente.

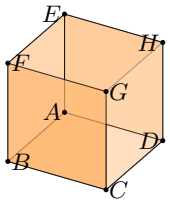

Fig. 1. Cuadríca base

Los puntos etiquetados previamente de la figura 1 denotados como A, B, C, D, E, F, G, H representan algunos de los puntos que satisfacen esta ecuación. Estos puntos son las esquinas del cubo en el espacio proyectivo y se representan en forma de matriz de coordenadas homogéneas:

$$A : [0, 0, 0, 1]$$
$$B : [1, 0, 0, 1]$$
$$C : [1, 1, 0, 1]$$
$$D : [0, 1, 0, 1]$$
$$E : [0, 0, 1, 1]$$
$$F : [1, 0, 1, 1]$$
$$G : [1, 1, 1, 1]$$
$$H : [0, 1, 1, 1]$$

Cada uno de estos puntos se representa como un vector de coordenadas homogéneas con un cuarto elemento igual a 1. Estos vectores de coordenadas homogéneas se utilizan comúnmente en geometría proyectiva para representar puntos en el espacio proyectivo [6].

A partir de esos puntos, se definen mas figuras para simular el juego, ver figura 2 que ilustra la linea formada por cubos.

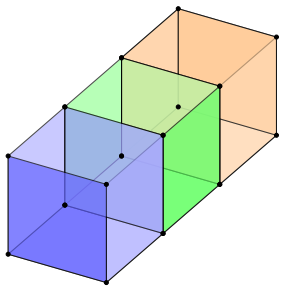

Fig. 2. Línea formada por cubos

La unión de cubos mediante cuádricas da lugar a una variedad de figuras que pueden ser controladas y definidas a partir de su geometría. Se pueden apreciar en la figura 3.

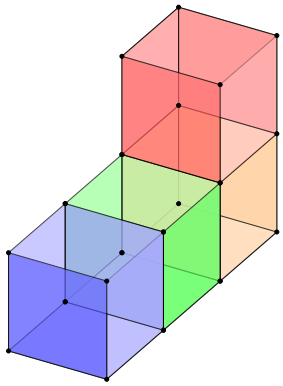

Fig. 3. **L** formada por cubos

Desde una perspectiva geométrica proyectiva, se resalta la importancia de la proyección en la imagen, donde se debe considerar minuciosamente la representación de objetos tridimensionales en un plano bidimensional. Un componente esencial en este proceso involucra la inclusión de un punto de fuga en la composición visual. Este punto de fuga permite que los objetos converjan hacia el horizonte y se fusionen en el infinito, contribuyendo así a crear una representación precisa y detallada de la escena en la imagen bidimensional. Consulte la figura 4, que muestra una de las figuras previamente generadas y cómo interactúan con el cubo. Estas figuras deben tener la capacidad de rotar a través de su matriz de rotación.

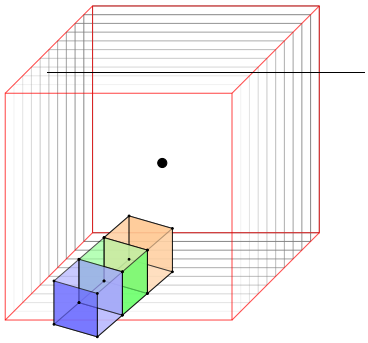

Fig. 4. Cubo con Punto de Fuga para Proyección en 2D

La rotación de un objeto 3D se puede representar como una transformación que conserva las líneas y los puntos. En otras palabras, una rotación no cambia la relación entre los puntos en un objeto.

Matemáticamente, una rotación de un objeto 3D se puede representar mediante una matriz de rotación. Esta matriz tiene la propiedad de que, si se multiplican las coordenadas de un punto por la matriz de rotación, las coordenadas del punto transformado seguirán estando en la misma línea que las coordenadas del punto original.

Por ejemplo, si tiene un objeto 3D representado por un conjunto de puntos, $\{P_1, P_2, \ldots, P_n\}$, el objeto rota alrededor del eje x en un ángulo θ mediante la siguiente matriz de rotación:

$$R = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix}$$

Al multiplicar las coordenadas de cada punto en el objeto por esta matriz de rotación, obtendremos las coordenadas del punto transformado. Es decir, si las coordenadas del punto P_1 son $(1, 2, 3)$, las coordenadas del punto P_1 transformado serán:

$$P_1' = R \cdot P_1 = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos(\theta) & -\sin(\theta) \\ 0 & \sin(\theta) & \cos(\theta) \end{bmatrix} \cdot \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 3 \end{bmatrix}$$

Lo anterior evidencia que las coordenadas del punto P_1 transformado siguen estando en la misma línea que las coordenadas del punto P_1 original.

En general, se puede usar cualquier matriz de rotación para rotar un objeto 3D. Sin embargo, las matrices de rotación que se utilizan en geometría proyectiva tienen la propiedad adicional de que conservan los puntos en el infinito.

Un punto en el infinito es un punto que se encuentra en el límite del espacio proyectivo, estos puntos se pueden representar mediante líneas. [7]

Si se tiene una línea en el espacio 3D, se puede representar como un conjunto de puntos que se encuentran en la misma recta. Si la línea rota alrededor del eje x en un ángulo θ, la matriz de rotación correspondiente tendrá la propiedad de que al multiplicar las coordenadas de cualquier punto en la línea por la matriz de rotación, las coordenadas del punto transformado seguirán estando en la misma línea que las coordenadas del punto original.

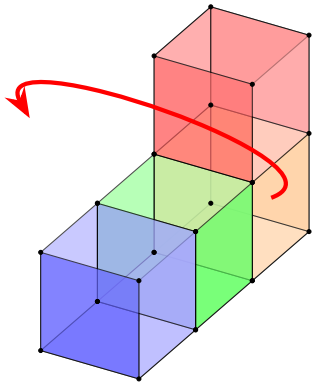

Fig. 5. Rotación

Respecto a la matriz de traslación, se utiliza para desplazar la figura tridimensional (3D) en el espacio proyectivo sin cambiar su orientación ni su forma. Las coordenadas de un punto tridimensional P en el espacio se representan generalmente como un vector de homogeneidad, que es un vector de cuatro elementos $[X, Y, Z, W]$, donde W es una coordenada homogénea que permite representar puntos en el infinito y simplificar ciertas transformaciones.

La matriz de traslación se representa como una matriz 4×4. La forma general de esta matriz de traslación en la dirección x, y y z se ve así:

$$T = \begin{bmatrix} 1 & 0 & 0 & t_x \\ 0 & 1 & 0 & t_y \\ 0 & 0 & 1 & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix}$$

Donde t_x, t_y, y t_z son las cantidades de traslación en las direcciones x, y, y z respectivamente.

Cuando esta matriz de traslación T se aplica a un punto P en coordenadas homogéneas $[X, Y, Z, W]$, se obtiene un nuevo punto P' en coordenadas homogéneas de la siguiente manera:

$$\begin{bmatrix} X' \\ Y' \\ Z' \\ W' \end{bmatrix} = T \cdot \begin{bmatrix} X \\ Y \\ Z \\ W \end{bmatrix}$$

Donde:

$$X' = X + t_x \cdot W$$

$$Y' = Y + t_y \cdot W$$

$$Z' = Z + t_z \cdot W$$

$$W' = W$$

Esta fórmula describe cómo se traslada un punto 3D en el espacio proyectivo. Es importante notar que el cuarto componente, W, no se modifica durante la traslación. Esto es esencial para mantener la consistencia en la representación de puntos en el infinito y asegurar que las transformaciones se realicen correctamente en el espacio proyectivo.

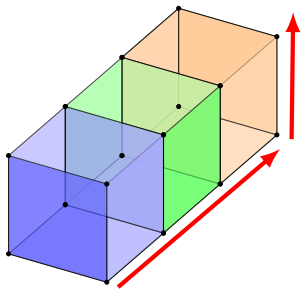

Fig. 6. Traslación

IV. METODOLOGÍA

Para la proyección de las figuras de $\mathbb{P}^3$ a $\mathbb{P}^2$, se implementó programación orientada a objetos en Python, utilizando librerías como numpy y matplotlib.

La estructura del código contiene 18 clases, con sus respectivos métodos, todo esto con el objetivo de controlar la traslación y rotación de las 3 figuras definidas anteriormente. (4,2,3)

A continuación se explicará brevemente las funciones que contienen cada clase:

A. *Definición de las figuras*

- Class Cube: Esta clase contiene un método estático, el cual define los vértices que conforman cada cara del cubo.
- Class Room: contiene la función que define el espacio y las dimensiones del cuarto donde serán rotadas y trasladadas las figuras diseñadas.
- Class LineShape: Aquí se define la función que forma la figura en línea de 3 cubos que se encuentran unidos verticalmente.
- Class Lshape: En esta clase se crea una función que utiliza la primera clase creada llamada Cube, para formar la figura L conformada por 4 cubos. La función permite alinear 3 cubos verticalmente, y ubicar un cubo adicional al lado derecho del tercer cubo.

B. *Implementación de movimiento a las figuras*

- Class FigureController: esta clase contiene un método que permite que cada figura pueda moverse, es decir, asigna una letra del teclado a determinado movimiento de la figura. Los movimientos definidos son de rotación y traslación, dónde:
  - 'w' permite la traslación positiva en el eje z
  - 'x' permite la traslación negativa en el eje z
  - 'a' permite la traslación negativa en el eje x
  - 'd' permite la traslación positiva en el eje x
  - 'e'permite la traslación positiva en el eje y
  - 'b' permite la traslación negativa en el eje y
  - 'u' permite la rotación positiva en el eje x
  - 'j' permite la rotación negativa en el eje x
  - 'i' permite la rotación positiva en el eje y
  - 'n' permite la rotación negativa en el eje y
  - 'p' permite la rotación positiva en el eje z
  - 'm' permite la rotación negativa en el eje z

C. *Transformación de matrices*

Para las transformaciones requeridas se definieron las siguientes funciones:

- translation_matrix: esta funcion crea una matriz de traslación.
- rotation_matrix: aquí se crea una matriz de rotación alrededor de un eje en específico.
- apply_transformation: aplica una transformación matricial a una figura.

D. *Aplicación de las clases*

Finalmente en un archivo llamado *main.py*, se encuentran importadas las clases creadas para realizar cada acción necesaria.

En este script, se definen las siguientes funciones:

- dibujar_cubo: esta función forma las aristas entre los vértices definidos para formar el cubo.
- dibujar_cuarto: esta función forma las aristas entre los vértices definidos para formar el cuarto donde se pueden trasladar y rotar las figuras.
- dibujar_lineshape: aquí se unen las vértices de los 3 cubos, que se encuentran unidos de manera vertical.
- dibujar_lshape: aquí se unen las vértices de los 3 cubos, que se encuentran unidos de manera vertical, con los vértices del cubo que se encuentra unido al tercer cubo en la parte derecha.
- on_key: esta función tiene como parámetro 'evento'. Aquí se hace usos de 3 funciones que permiten actualizar las posiciones de cada una de las figuras, y a su vez, asigna un número del teclado a la figura que el usuario quiera controlar en el momento, así:
  - '1': se asigna a la figura del cubo.
  - '2': se asigna a la figura en forma de L.
  - '3': se asigna a la figura que forma una línea de cubos.

V. ANÁLISIS Y RESULTADOS

Un análisis exhaustivo del cuarto implica examinar su dimensionalidad y su representación en la figura 7. En esta vista lateral del proyecto, se detallan las figuras presentes en el entorno junto con sus respectivos grados de libertad. En este contexto, las figuras tienen la capacidad de realizar movimientos y rotaciones en todas las direcciones disponibles, lo que contribuye a la versatilidad y complejidad del proyecto.

En la figura 8, se observa el resultado que se obtiene al ejecutar todos los scripts que conforman la programación para lograr controlar la rotación y la traslación de las figuras en 3d ubicadas en un cuarto de juego definido previamente. Cada figura puede controlarse al presionar la tecla asignada, como anteriormente se había mencionado.

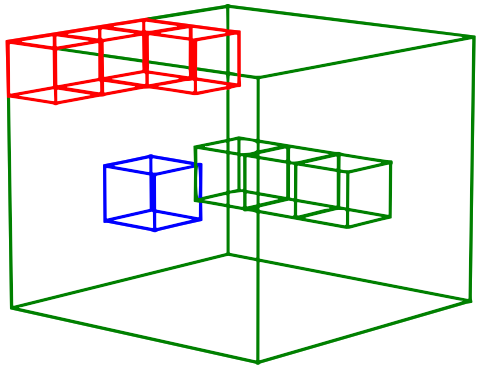

Fig. 7. Vista lateral

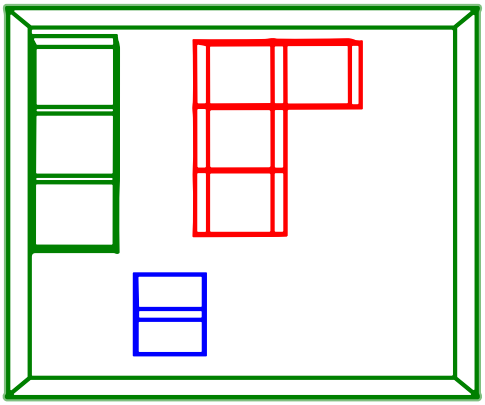

Fig. 8. Figuras en 3D

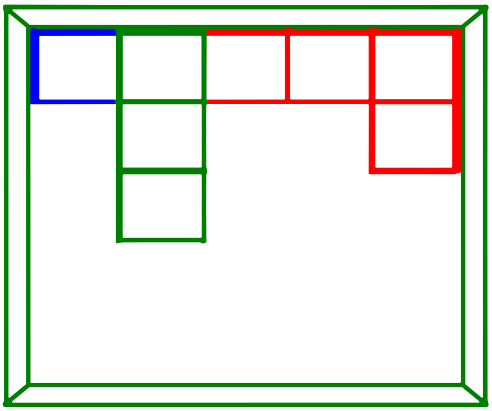

Fig. 9. Figuras ubicadas en el cuarto de juego

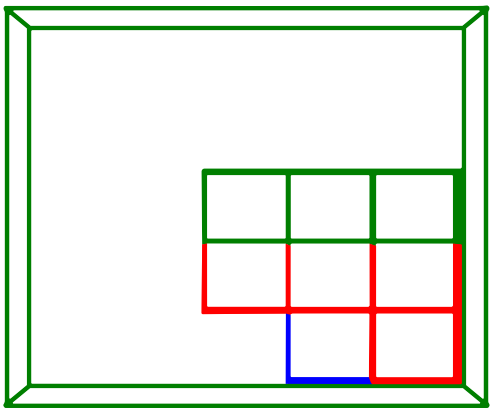

Fig. 10. Figuras ubicadas en el cuarto de juego

Como se observa en la figura 9, las 3 figuras pueden ser controladas en rotación y traslación mediante el teclado, de tal forma que puedan ser ubicadas adecuadamente en el cuarto de juego. Para esto se tuvo que colocar estático el cuarto de juego, con el fin de que al rotar las figuras, no se latera su forma ni su ubicación en el espacio.

Lo anterior prueba que si hay una correcta proyectividad de $\mathbb{P}^3$ a $\mathbb{P}^2$, ya que las figuras a pesar de estar en 3 dimensiones, se pueden visualizar en 2 dimensiones de manera correcta y conservando la forma que fue definida en la primera clase.

En contraste con la figura 9, en la figura 10 se observa como las figuras pueden cambiar de orientación, modificado sus rotaciones y traslaciones. La figura verde que en principio se observaba de manera vertical, ahora se ubica de manera horizontal, por elección del usuario que se encuentra controlando las figuras del juego.

VI. Conclusiones

- La aplicación de los conocimientos matemáticos sobre geometría proyectiva, fueron fundamentales a la hora de plantear las diversas ecuaciones y funciones que conformaron cada método de cada clase de la programación desarrollada para la visualización de las piezas y el cuarto de juego.
- Desde la perspectiva de la geometría proyectiva, se ha explorado la complejidad y versatilidad del espacio proyectivo, haciendo hincapié en las coordenadas homogéneas y las transformaciones intrínsecas que ellas posibilitan. Al introducir una dimensión adicional, estas coordenadas no sólo brindan una representación unificada de puntos finitos e infinitos, sino que también simplifican las operaciones matemáticas al evitar tratamientos especiales.
- Los conocimientos de geometría proyectiva permiten entender cómo se proyectan los objetos del mundo real en imágenes. Además permite modelar las distorsiones que se producen en las imágenes debido a la perspectiva. Todo lo anterior es útil a la hora de aplicarse en visión por computador, como en el reconocimiento de objetos, seguimiento de objetos y calibración de cámaras.

References

[1] Stefanelli. Proyecciones ortogonales. [Online]. Available: https://www.stefanelli.eng.br/es/proyecciones-ortogonales-1/

[2] Different view. [Online]. Available: https://nzmaths.co.nz/resource/different-view
[3] The method of least squares. [Online]. Available: https://math.libretexts.org/Bookshelves/Linear_Algebra/Interactive_Linear_Algebra_(Margalit_and_Rabinoff)/06%3A_Orthogonality/6.5%3A_The_Method_of_Least_Squares
[4] R. van den Boomgaard. (2016-2017) Projective geometry in 3d. [Online]. Available: https://staff.fnwi.uva.nl/r.vandenboomgaard/IPCV20162017/LectureNotes/CV/PinholeCamera/ProjectiveGeometry.html#projective-geometry-in-3d
[5] Stefanelli. Proyecciones ortogonales. [Online]. Available: https://www.stefanelli.eng.br/es/proyecciones-ortogonales-1/
[6] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. Cambridge, MA: MIT Press, 2016.
[7] R. Szeliski, *Computer Vision: Algorithms and Applications*. London: Springer, 2010.