Por ejemplo, el usuario puede agregar materiales con el comando *addMaterial()*. A medida que el usuario comienza a escribir comandos en la terminal, los comandos disponibles aparecen a un costado de la misma. El usuario puede usar la tecla *shift* para seleccionar cada una de las opciones disponibles. Una vez haya seleccionado la opción buscada, el usario puede usar la tecla *enter* para hacer aparecer el comando en la terminal (véase la figura **5-9**).

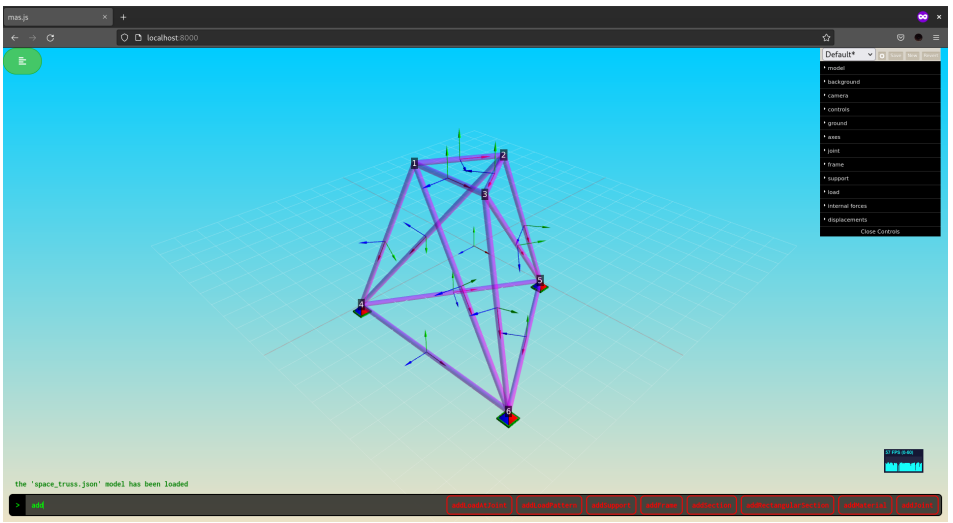

Figura 5-9: Opciones disponibles de mas.js a medida que se escribe en la terminal.

Las funciones para agregar objetos reciben los mismos parámetros de entrada que su contraparte en pymas. Por ejemplo, para agregar un material con módulo de elasticidad igual a 2100 t/cm2, el usuario puede escribir *addMaterial(material1, 21000000)*.

El usuario puede ver el contenido del modelo con la función *getStructure()*, la cual imprime el archivo. En la figura **5-10** se presenta mas.js después de haber agregado un material.

Así mismo, el usuario puede agregar otros objetos al modelo de la misma manera. Por ejemplo, en la figura **5-11** se presenta el modelo después de agregar los nodos de la estructura. Por ejemplo, para agregar el nodo 3, el usuario puede ejecutar *addJoint(3, 5.25, 6, 4.8)*.

Una vez el usuario ha definido los materiales, las secciones transversales y los nodos de la estructura, puede agregar los elementos aporticados con la función *addFrame*, muy similar a como se hace en pymas. En la figura **5-12** se presenta el modelo después de

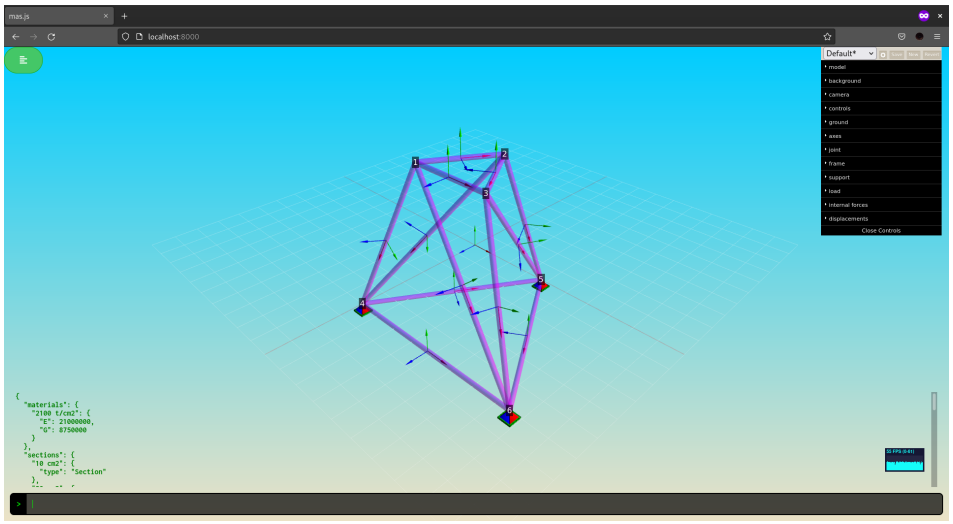

Figura 5-10: Contenido del modelo al ejecutar la funcion `getStructure()`.

agregar los elementos. Por ejemplo, para agregar el elemento 1-3, el usuario puede ejecutar el comando *addFrame(1-3, 1, 3, section, material)*.

Así mismo, el usuario puede agregar los apoyos de la estructura con la función *addSupport()*. En la figura **5-13** se presenta el modelo después de agregar los apoyos de la estructura. Por ejemplo, para agregar el apoyo del nodo 6, el usuario puede ejecutar el comando *addSupport(4, True, True, True, False, False, False)*.

Finalmente, el usuario puede agregar cargas puntuales, una vez haya agregado patrones de carga. Para agregar un patrón de carga, el usuario puede ejecutar el comando *addLoadPattern*. Para agregar una carga puntual, el usuario puede ejecutar el comando *addLoadAtJoint*. En la figura **5-14** se presenta el modelo después de agregar las cargas puntuales.

El usuario puede ejecutar una vez más la función `getStructure()` para ver el contenido del modelo de la estructura. A continuación se presenta el contenido del modelo de la cercha tridimensional.


```
{
    "materials": {
        "2100 t/cm2": {
            "name": "2100 t/cm2",
            "E": 21000000.0,
            "G": 8750000
        }
    },
    "sections": {
        "10 cm2": {
            "type": "Section",
            "name": "10 cm2",
```

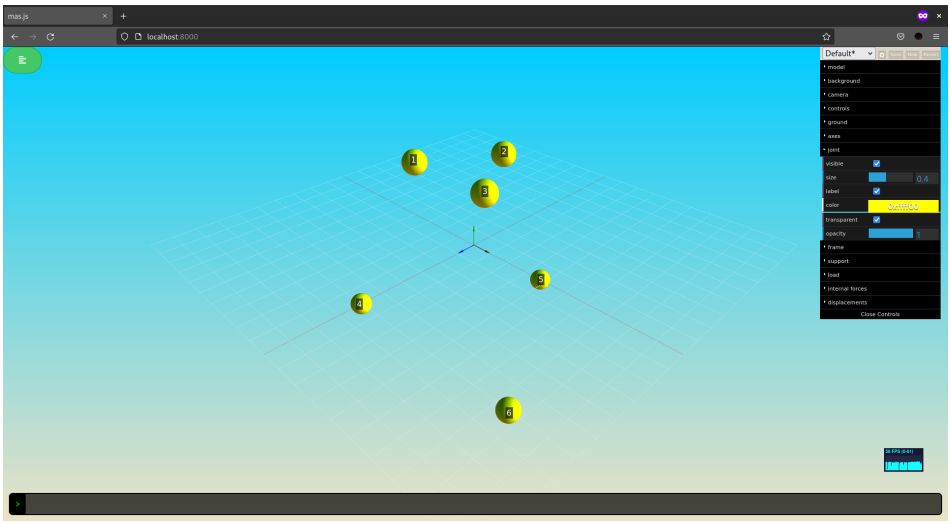

Figura 5-11: Modelo en mas.js después de agregar los nodos de la estructura.


```
        "A": 0.001,
        "J": 0.000000000001,
        "Iy": 0.000000000001,
        "Iz": 0.000000000001
    },
    "20 cm2": {
        "type": "Section",
        "name": "20 cm2",
        "A": 0.002,
        "J": 0.000000000001,
        "Iy": 0.000000000001,
        "Iz": 0.000000000001
    },
    "40 cm2": {
        "type": "Section",
        "name": "40 cm2",
        "A": 0.004,
        "J": 0.000000000001,
        "Iy": 0.000000000001,
        "Iz": 0.000000000001
    },
    "50 cm2": {
        "type": "Section",
        "name": "50 cm2",
        "A": 0.005,
        "J": 0.000000000001,
        "Iy": 0.000000000001,
        "Iz": 0.000000000001
    }
},
"joints": {
    "1": {
        "name": "1",
        "x": 2.25,
        "y": 6,
        "z": 4.8
    },
    "2": {
        "name": "2",
        "x": 3.75,
        "y": 6,
        "z": 2.4
    },
    "3": {
        "name": "3",
        "x": 5.25,
        "y": 6,
        "z": 4.8
    },
    "4": {
        "name": "4",
        "x": 0.0,
        "y": 0,
        "z": 6.0
    },
    "5": {
        "name": "5",
        "x": 3.75,
        "y": 0,
        "z": 0.0
    },
    "6": {
        "name": "6",
        "x": 7.5,
        "y": 0,
        "z": 6.0
    }
},
```

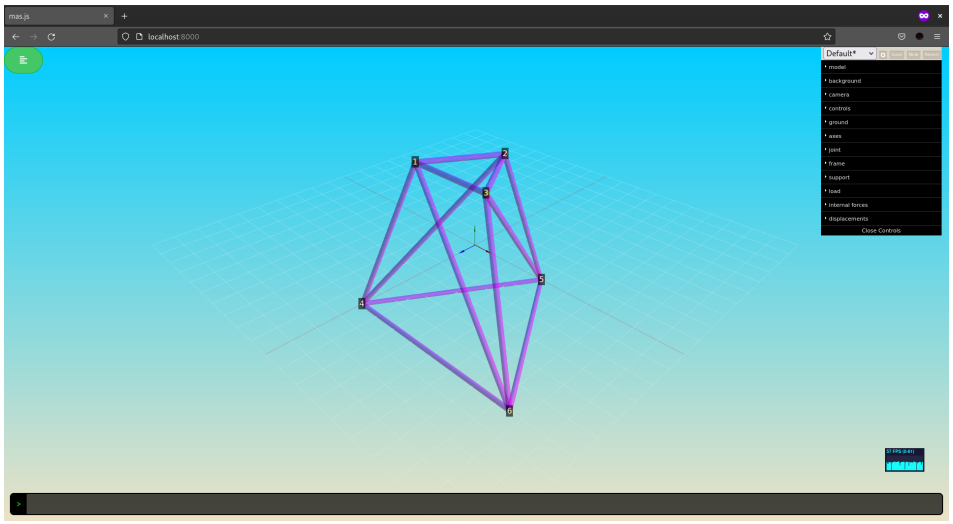

Figura 5-12: Modelo en mas.js después de agregar los elementos aporticados de la estructura.


```
"frames": {
    "1-2": {
        "name": "1-2",
        "joint_j": "1",
        "joint_k": "2",
        "material": "2100 t/cm2",
        "section": "20 cm2"
    },
    "1-3": {
        "name": "1-3",
        "joint_j": "1",
        "joint_k": "3",
        "material": "2100 t/cm2",
        "section": "20 cm2"
    },
    "1-4": {
        "name": "1-4",
        "joint_j": "1",
        "joint_k": "4",
        "material": "2100 t/cm2",
        "section": "40 cm2"
    },
    "1-6": {
        "name": "1-6",
        "joint_j": "1",
        "joint_k": "6",
        "material": "2100 t/cm2",
        "section": "50 cm2"
    },
    "2-3": {
        "name": "2-3",
        "joint_j": "2",
        "joint_k": "3",
        "material": "2100 t/cm2",
        "section": "20 cm2"
    },
    "2-4": {
        "name": "2-4",
        "joint_j": "2",
        "joint_k": "4",
        "material": "2100 t/cm2",
        "section": "50 cm2"
    },
    "2-5": {
        "name": "2-5",
        "joint_j": "2",
        "joint_k": "5",
        "material": "2100 t/cm2",
        "section": "40 cm2"
    },
    "3-5": {
        "name": "3-5",
        "joint_j": "3",
        "joint_k": "5",
        "material": "2100 t/cm2",
        "section": "50 cm2"
    },
    "3-6": {
        "name": "3-6",
        "joint_j": "3",
        "joint_k": "6",
        "material": "2100 t/cm2",
        "section": "40 cm2"
    },
    "4-5": {
        "name": "4-5",
```

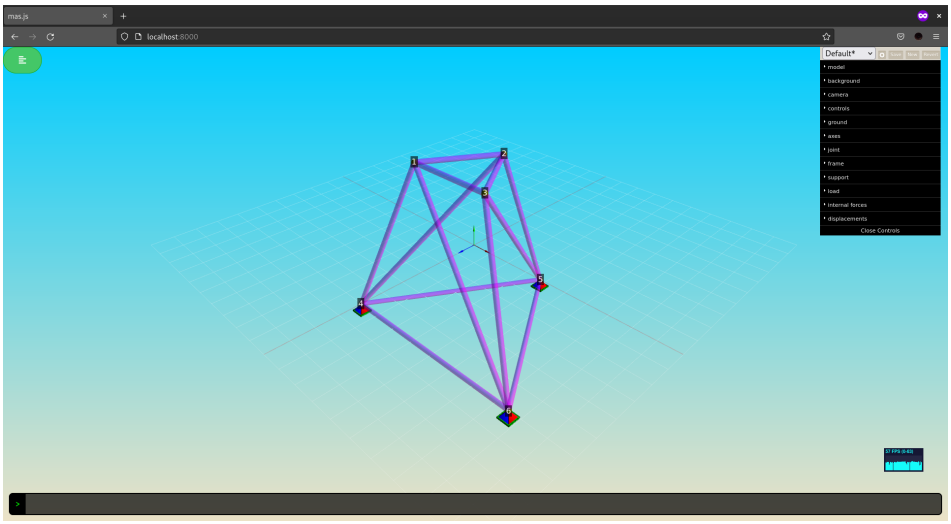

Figura 5-13: Modelo en mas.js después de agregar los apoyos de la estructura.


```
        "joint_j": "4",
        "joint_k": "5",
        "material": "2100 t/cm2",
        "section": "10 cm2"
    },
    "4-6": {
        "name": "4-6",
        "joint_j": "4",
        "joint_k": "6",
        "material": "2100 t/cm2",
        "section": "10 cm2"
    },
    "5-6": {
        "name": "5-6",
        "joint_j": "5",
        "joint_k": "6",
        "material": "2100 t/cm2",
        "section": "10 cm2"
    }
},
"supports": {
    "4": {
        "joint": "4",
        "ux": true,
        "uy": true,
        "uz": true
    },
    "5": {
        "joint": "5",
        "ux": true,
        "uy": true,
        "uz": true
    },
    "6": {
        "joint": "6",
        "ux": true,
        "uy": true,
        "uz": true
    }
},
"load_patterns": {
    "point loads": {
        "name": "point loads",
        "joints": {
            "1": [
                {
                    "load_pattern": "
point loads",
                    "joint": "1",
                    "fx": 10,
                    "fy": 15,
                    "fz": -12
                }
            ],
            "2": [
                {
                    "load_pattern": "
point loads",
                    "joint": "2",
                    "fx": 5,
                    "fy": -3,
                    "fz": -10
                }
            ],
            "3": [
                {
                    "load_pattern": "
point loads",
```

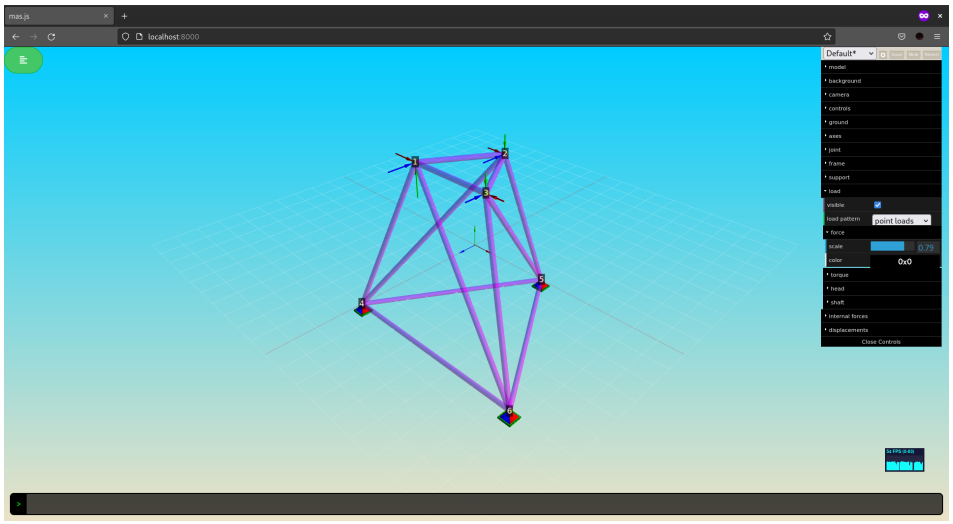

Figura 5-14: Modelo en mas.js después de agregar las cargas puntuales.

```
        "joint": "3",
        "fx": -4,
        "fy": -2,
        "fz": -6
    }
```

```
                ]
            }
        }
    }
}
```

Una vez se ha modelado la estructura, el usuario puede guardar la información en un archivo de texto para analizarla con pymas. En la carpeta `04_mas.js` del repositorio pymas hay un ejemplo para analizar esta estructura. La información del modelo se guardó en un archivo llamado `space_truss.json`, para ser analizada con el programa `mas.js.py`. En el algoritmo 5.5 se presentan las instrucciones de este programa.

Algoritmo 5.5: Algoritmo del archivo `mas.js.py` para analizar el archivo `space_truss.json`, creado con el programa mas.js.

```
import makepath
from pymas import Structure

# create a model
model = Structure(True, True, True)

# open model
model.open('space_truss.json')

# solve model
model.solve()
```

```
# export model
model.export('space_truss-solved.json')
```

Inicialmente, el programa importa el módulo `makepath` para indicarle al intérprete de Python donde buscar el código fuente del programa pymas. Seguidamente, el programa importa la clase `Structure` del módulo `pymas`, para analizar el archivo `space_truss.json`.

Una vez se ha importado la clase `Structure`, se crea un objeto para analizar únicamente las traslasiones de los nodos de la estructura. Esto se hace pasándole al constructor de la clase `Structure` tres valores `True`, para cada una de las posibles traslaciones.

Finalmente, se abre el archivo `space_truss.json` con el método `open()` de la clase `Structure`, para leer la información del modelo y agregarla a la variable `structure`, se soluciona con el método `solve()` y se guarda el análisis de la estructura en el archivo `space_truss-solved.json`.

En la tabla **5-3** se presentan los desplazamientos de los nodos de la estructura.

Nodo	1	2	3
u_x [m]	8.058×10^{-4}	2.226×10^{-3}	7.512×10^{-4}
u_y [m]	3.326×10^{-5}	-7.277×10^{-4}	3.670×10^{-4}
u_z [m]	-4.464×10^{-3}	-2.732×10^{-3}	-1.773×10^{-3}

Tabla 5-3: Desplazamientos en los nodos del modelo para el patrón de carga *point loads*.

En la tabla **5-4** se presentan las reacciones de los apoyos de la estructura.

Nodo	4	5	6
f_x [m]	−15.900	1.700	3.200
f_y [m]	−30.800	27.120	−6.340
f_z [m]	13.120	13.600	1.280

Tabla 5-4: Reacciones de los apoyos del modelo para el patrón de carga *point loads*.

El programa `mas.js.py` guarda el análisis de la estructura en un archivo llamado `space_truss.json`. Este archivo se puede abrir en mas.js para ver los resultados del

análisis. En la figura **5-15** se presenta la fuerza axial debido a las cargas externas.

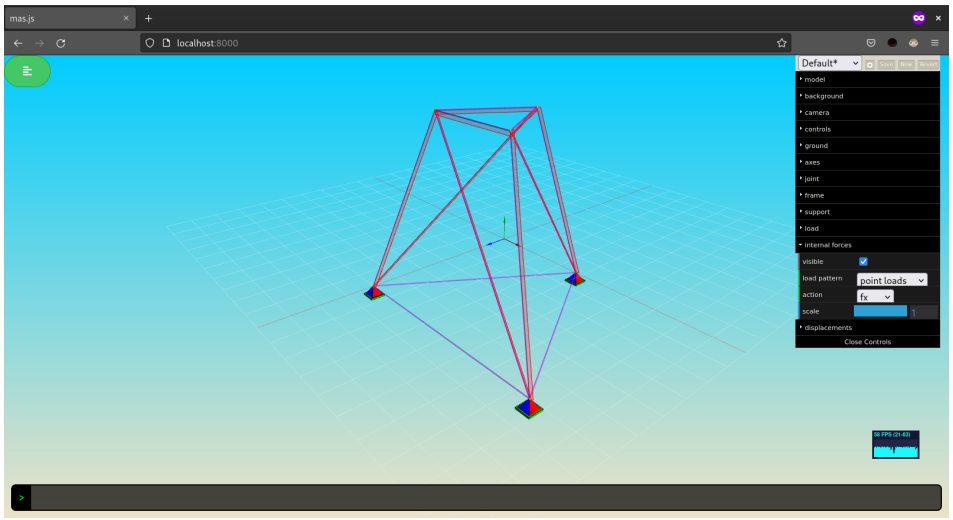

Figura 5-15: Fuerza axial de los elementos de la cercha tridimensional sujetas a las cargas externas.

6 Conclusiones

En esta tesis de maestría se desarrollaron los programas a código abierto pymas y mas.js, capaces de analizar estructuras aporticadas tridimensionales bajo cargas estáticas, considerando un comportamiento lineal y elástico del material.

El primer programa es una librería de Python que implementa el método directo de rigideces. Con esta librería se pueden analizar estructuras aporticadas tridimensionales sometidas a diferentes patrones de carga. El segundo es un programa de computador (aplicación web) para crear y revisar los modelos estructurales en un navegador de internet. El programa muestra una escena tridimensional creada con la librería *Three.js*. Al mismo tiempo modifica un archivo en formato JSON en el cual se almacena la información geométrica y diferentes características del modelo.

pymas y mas.js se desarrollaron para utilizar el mismo esquema de archivos para almacenar la información del modelo, con archivos de formato JSON. De esta manera pymas puede resolver los modelos creados con mas.js y mas.js puede interpretar la solución calculada por pymas. Este formato de archivo de transferencia permite que cualquier persona pueda leerlo y entender fácilmente la información almacenada.

Ambos desarrollos tienen licencia MIT, por lo que sus usuarios van a tener el menor número de restricciones al momento de usar, revisar, mejorar y distribuir los programas desarrollados. Se espera que esto impacte positivamente en el desarrolllo del cógido, al tomar en cuenta la retroalimentación y aportes de sus usuarios.

pymas puede funcionar como un programa de computador aislado para analizar estructuras aporticadas. Pero, al ser una librería de Python en la que se implementa el método directo de rigideces, se facilita su uso como motor de análisis numérico de otros desarrollos.

La implementación estándar de Python está escrita en ANSI C, por lo que puede ser compilada en cualquier plataforma y sistema operativo actual. En consecuencia, pymas puede ser ejecutado en diferentes dispositivos, desde supercomputadores, pasando por

los computadores convencionales, hasta los teléfonos inteligentes.

Este lenguaje de programación cuenta con SciPy, lo que hace sencillo trabajar con matrices en Python. Esto facilita el desarrollo de pymas, ya que sólo hay que ocuparse de implementar los algoritmos propios del método directo de las rigideces y no del álgebra lineal. Adicionalmente, debido a la simplicidad de la sintaxis del lenguaje, las líneas de código resultan fáciles de leer.

Las clases desarrolladas y su funcionamiento están pensadas para facilitar la expansión de nuevas funcionalidades de la librería pymas, como por ejemplo análisis dinámicos. Sin embargo, dichas clases y definiciones podrán cambiar según los nuevos desarrolladores lo consideren apropiado y, con el tiempo, tomarán su forma definitiva.

Por su parte, mas.js está desarrollado con JavaScript, que junto a HTML y CSS conforman las tecnologías web. Esto abre la posibilidad de poder ejecutar mas.js nativamente en diferentes dispositivos, y no solo a través de un computador con el programa almacenado localmente.

Este programa se ve muy prometedor y desde ya tiene muchos retos por delante para satisfacer todas las necesidades de sus usarios. Por lo pronto se verificó que con la librería *Three.js* es posible desarrollar la herramienta de pre y posproceso necesaria para poder modelar las estructuras aporticadas y revisar los resultados. No obstante, un desarrollo completo de este tipo merece la pena ser el tema principal de otro trabajo, ya que esta es un área del conocimiento agena al análisis estructural.

Referencias

Computers & Structures. (2017). *CSi Anlysis Reference Manual*.

Computers & Structures. (2021). ETABS System Requirements [Accedido: 2021-10-11].

Dirksen, J. (2015). *Learning Three.js–the JavaScript 3D library for WebGL : create stunning 3D graphics in your browser using the Three.js JavaScript library*. Packt Publishing.

Dunn, F. (2002). *3D math primer for graphics and game development*. Wordware Pub.

EERI. (2016). *Connections. Edward L. Wilson*. Earthquake Engineering Research Institute.

Fenves, S. (1965). *STRESS: A Reference Manual. A Problem-Oriented Computer Language for Structural Engineering*. The M.I.T Press.

Gere, J. (1986). *Mecánica de materiales*. Grupo Editorial Iberoamérica.

Harris, C. R., Millman, K. J., van der Walt, S. J., Gommers, R., Virtanen, P., Cournapeau, D., Wieser, E., Taylor, J., Berg, S., Smith, N. J., Kern, R., Picus, M., Hoyer, S., van Kerkwijk, M. H., Brett, M., Haldane, A., del R'ıo, J. F., Wiebe, M., Peterson, P., … Oliphant, T. E. (2020). Array programming with NumPy. *Nature*, *585*(7825), 357-362. https://doi.org/10.1038/s41586-020-2649-2

Kassimali, A. (2011). *Matrix Analysis of Structures* (2.ª ed.). Cengage learning.

Lutz, M. (2013). *Learning Python*. O'Reilly.

MDN. (2021). Window.requestAnimationFrame(). Consultado el 8 de marzo de 2021, desde https://developer.mozilla.org/en-US/docs/Web/API/window/requestAnimationFrame

Overflow, S. (2020, 27 de mayo). Stack Overflow Developer Survey 2020. Consultado el 18 de agosto de 2021, desde https://insights.stackoverflow.com/survey/2020/

Reddy, J. N. (1993). *An introduction to the finite element method*. McGraw-Hill.

Sten, J. (2022). *Formvlae generales pro translatione qvacvnqve corporvm rigidorvm*.

Three.js authors. (2021a). Object3D. Consultado el 8 de marzo de 2021, desde https://threejs.org/docs/#api/en/core/Object3D

Three.js authors. (2021b). Shape. Consultado el 12 de marzo de 2021, desde https://threejs.org/docs/#api/en/extras/core/Shape

Threejsfundamentals authors. (2021). Three.js Scene Graph. Consultado el 27 de marzo de 2021, desde https://threejsfundamentals.org/threejs/lessons/threejs-scenegraph.html

Uribe-Escamilla, J. (1995). *Microcomputadores en ingeniería estructural*. Universidad Nacional de Colombia y Ecoe Ediciones.

van Rossum, G. (2019). Welcome to Python.org [Accessed: 2019-03-15]. Consultado el 15 de marzo de 2019, desde https://www.python.org/

Virtanen, P., Gommers, R., Oliphant, T. E., Haberland, M., Reddy, T., Cournapeau, D., Burovski, E., Peterson, P., Weckesser, W., Bright, J., van der Walt, S. J., Brett, M., Wilson, J., Millman, K. J., Mayorov, N., Nelson, A. R. J., Jones, E., Kern, R., Larson, E., … SciPy 1.0 Contributors. (2020). SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. *Nature Methods*, *17*, 261-272. https://doi.org/10.1038/s41592-019-0686-2

Weaver, W. J. & Gere, J. (1990). *Matrix analysis of framed Structures*. Van Nostrand Reinhold.

Wilson, E. L. & Dovey, H. H. (1972). Three dimensional analysis of building systems - TABS. *Earthwuake engineering research center*.

Wilson, E. L., Hollings, J. P. & Dovey, H. (1975). Three dimensional analysis of building systems (extended version). *Earthwuake engineering research center*.