JCAAC (2024), **1**, 49–56

SOFTWARE

Caso de uso de scripting en Stellarium

Sergio Díaz Ruiz[1]

[1]Asociación Astronómica Astronomía Sevilla, Spain. E-mail: sergio.diaz.ruiz@gmail.com.

Keywords: Scripting, Stellarium

Este artículo adjunta un *software* accesible en
`https://github.com/JCAAC-FAAE/No01-Nov2024-Caso_de_uso_de_scripting_en_Stellarium`

Resumen
En nuestros proyectos de cálculo de fenómenos obtenemos como resultado final un conjunto de fechas, entre otros datos de interés. Lejos de terminar aquí, la siguiente tarea, no menos importante y que puede requerir un esfuerzo considerable, es la presentación de los resultados. En este artículo proponemos el uso de Stellarium para este fin. Para ello mostraremos cómo generar desde nuestro programa un script de Stellarium que empaquete las fechas de los fenómenos que hemos calculado. Si bien es posible entregar directamente este script al usuario para que lo ejecute sobre Stellarium, aquí exploramos la opción de que el script capture imágenes de los fenómenos tal como los representa Stellarium, que posteriormente podamos emplear para ilustrar un documento o página web.

Abstract
In our projects on the computation of phenomena, the final output is typically a set of dates along with other relevant data. However, this is far from the end of the process. An equally important and often time-consuming task is the presentation of these results. In this article, we propose using Stellarium for this purpose. We will demonstrate how to generate a Stellarium script from our program, which organizes the dates of the calculated phenomena. While it is possible to provide this script directly to the user to run in Stellarium, we will explore an alternative approach: using the script to capture images of the phenomena as depicted by Stellarium. These images can then be used to illustrate documents or web pages.

1. Introducción

En este artículo partimos de la situación en la que hemos completado el cálculo de cierto tipo de fenómenos, obteniendo una lista con las fechas de interés para la observación de un determinado objeto, por ejemplo, las fechas julianas en las que se producen fenómenos triples[1] de los satélites galileanos, dentro de un intervalo temporal dado.

Queremos crear una visualización de estos fenómenos sin tener que implementar por nuestra cuenta el código para representar gráficamente Júpiter con sus satélites y las sombras que éstos proyectan; en lugar de ello, podemos escribir un script Stellarium que genere una animación[2] o, como detallaremos en este artículo, que haga capturas de pantalla en los instantes de interés, como la mostrada en la figura 1.

Proponemos una plantilla de script Stellarium para visualización de los fenómenos que esencialmente se encarga de centrar el objeto principal y recorrer las lista de fechas de interés capturando la imagen

[1]En los fenómenos "triples" de los satélites galileanos se producen simultáneamente al menos tres fenómenos de tránsito y/o sombra sobre el disco de Júpiter. Las fechas julianas seleccionadas aquí corresponden al instante central del fenómeno. Los fenómenos triples entre 1981-2040 fueron calculados por el célebre Jean Meeus [1] y actualizados por el autor para el periodo 2025-2049 [2].

[2]La animación se puede capturar como vídeo usando ObsStudio, por ejemplo.

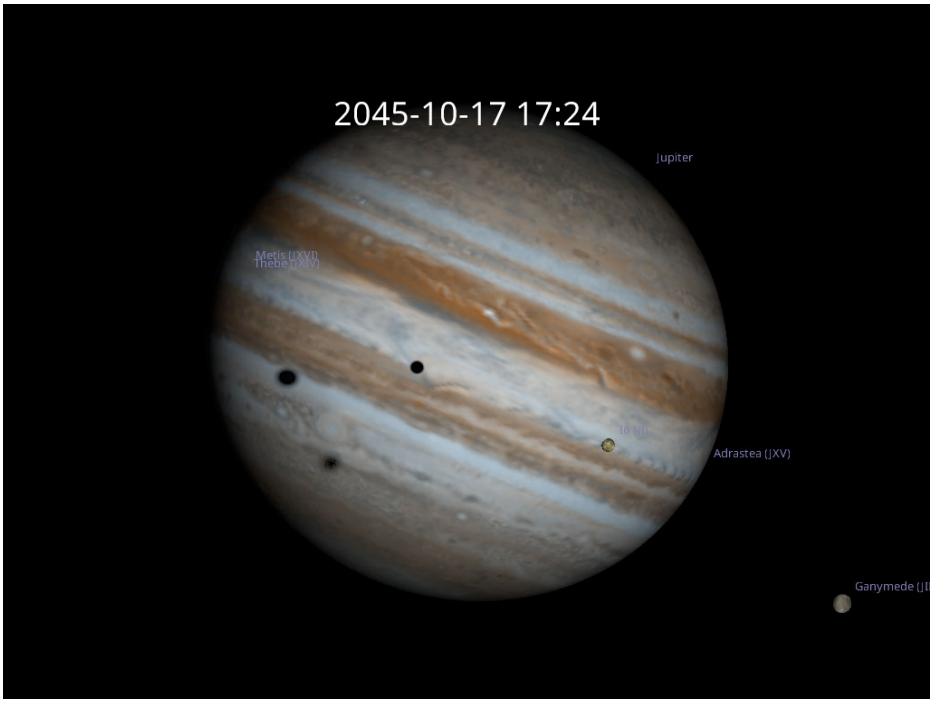

Figura 1. *Ejemplo de fenómeno "triple", en este caso cuádruple, al involucrar simultáneamente tres sombras y un tránsito. Los observadores situados en la parte sur de la banda ecuatorial y en torno al océano Índico serán los que estén mejor posicionados para observar este caso particular.*

generada por Stellarium para cada una de ellas. Al ser una plantilla, la podemos reutilizar en distintos proyectos, simplemente sustituyendo el objeto principal y las fechas de interés.

Si hemos generado la lista de fechas con nuestro propio código de cálculo, podemos extenderlo fácilmente para que también genere directamente el script Stellarium, ya que no es más que un fichero de texto plano. Mostramos un ejemplo de implementación, escrito en python, que podemos integrar en nuestro código para generar los scripts Stellarium a partir de la plantilla y los datos de los fenómenos a representar.

2. Scripting en Stellarium

Stellarium [3] contiene un intérprete de javascript compatible con la especificación ECMAScript 2016[3] [4], desde el cual podemos acceder a objetos y métodos con los que controlar la visualización. Basta conocer los conceptos básicos de programación y tener un poco de rodaje con lo más esencial de javascript [5]. Con ello, podemos seguir la sección 17, *Scripting*, de la guía oficial de Stellarium [6]. Cuando necesitemos encontrar qué método llamar y qué parámetros usar para realizar una cierta acción sobre Stellarium, debemos consultar la documentación de referencia del interfaz de programación (API, *Application Program Interface*) [7]. En este artículo cubriremos los métodos necesarios para visualizar los fenómenos de interés.

En Stellarium, pulsando F12 abrimos la consola de scripting, figura 2. En la pestaña "Script"se puede editar directamente el código, que se ejecutará al pulsar ►. El script se puede salvar en o cargar desde un fichero, que por defecto tendrá extensión `.ssc`. La consola presenta opciones adicionales, descritas en la guía [6].

[3] Las versiones recientes de Stellarium, en concreto a partir de la 22.3, están basadas en la librería Qt6 e integran el intérprete QJSEngine.

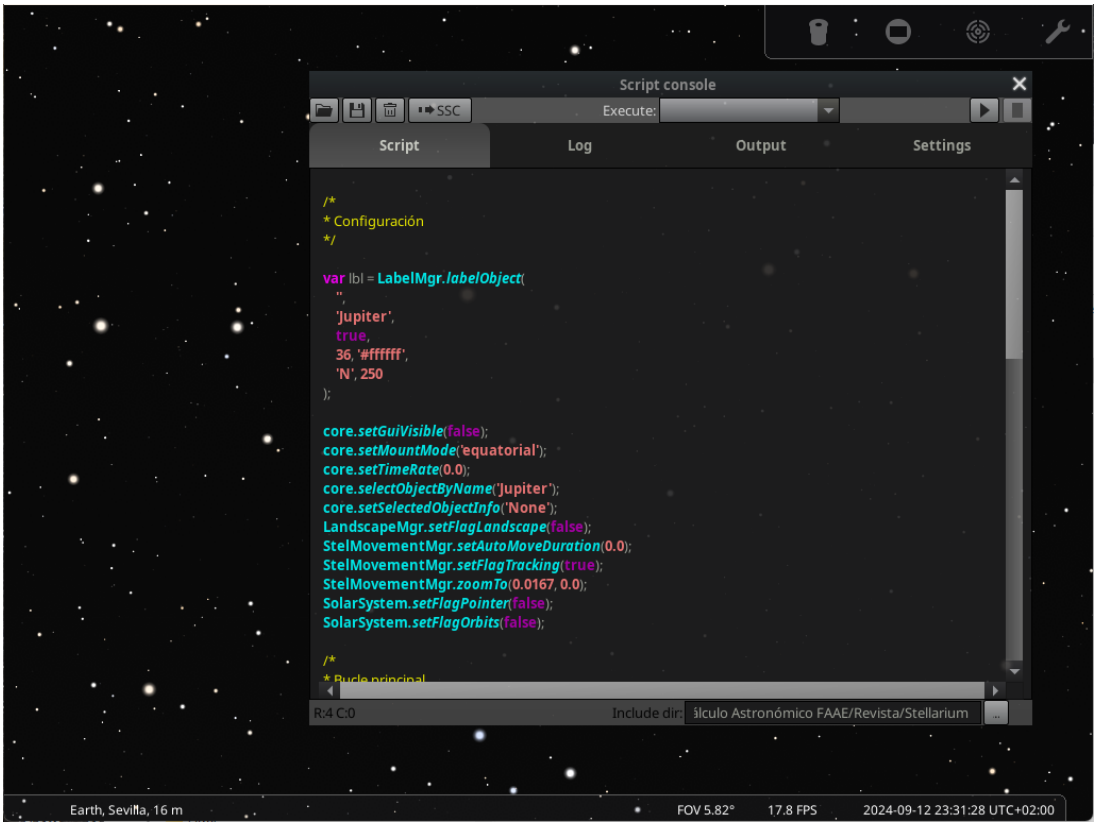

Figura 2. *Consola de scripting en Stellarium.*

3. Plantilla de script para visualización de fenómenos

El script que presentamos tiene tres secciones principales: *configuración*, donde seleccionamos el objeto y preparamos la presentación gráfica; *bucle principal*, donde se realiza el barrido por las fechas de interés, actualizando la vista principal; y *limpieza*, donde deshacemos algunos de los cambios anteriores de cara a devolverle el control al usuario.

El interfaz que nos permite interactuar con la funcionalidad principal de la aplicación es la clase `StelMainScriptAPI`, que ya está instanciada en el objeto `core`. Existen otras clases, tales como `LabelMgr`, `SolarSystem` o `StelMovementMgr`, usadas en este script, que ofrecen métodos estáticos (es decir, se pueden llamar sin instanciar la clase) más especializados para administrar etiquetas en la vista principal, configurar la vista de objetos de sistema solar o el movimiento de la "cámara", por citar algunos. La descripción de todas las clases y métodos disponibles está disponible en la documentación de referencia del API de scripting [7]. Por tanto, aunque el API de Stellarium está orientado a objetos, el uso que se hace de ellos es muy sencillo, puramente funcional: p.ej., en lugar de tener una función global `zoomTo()` para cambiar el campo de visión, disponemos del método `StelMovementMgr.zoomTo()`. La única excepción es el caso de `StelMainScriptAPI`, cuyos métodos deben ser invocados desde el objeto `core`, p.ej. `core.setDate()` para cambiar la fecha, en lugar de `StelMainScriptAPI.setDate()`.

El script para este caso de uso se comenta línea por línea en el recuadro de la figura 3. Como puede observarse, muchos de los valores literales se han dejado indicados como '`{parámetro}`', por lo que el código mostrado es realmente una plantilla y no un script que podamos copiar en la consola y ejecutar

directamente. Esto es intencionado, puesto que nuestro objetivo es generar el script final desde nuestro código, sustituyendo esos parámetros por los valores de interés sobre esta plantilla. Los parámetros son los siguientes:

- `script_name`: usado como prefijo en el nombre de los ficheros de captura de pantalla
- `object_name`: nombre del objeto a centrar, tal como lo indicaríamos en la barra de búsqueda de Stellarium
- `jd_list`: lista de fechas julianas, separadas por coma
- `fov_deg`: tamaño del campo de visión (FoV, *Field of View*), en grados
- `size`: tamaño de la fuente de la etiqueta en la que se mostrará la fecha y hora, en píxeles
- `color`: color de la etiqueta, en notación HTML, p.ej. '`#ffffff`' para blanco
- `side`: '`N`', '`S`', '`E`' o '`W`' para indicar la posición de la etiqueta respecto del objeto (encima, debajo, izquierda o derecha, respectivamente)
- `distance`: distancia de la etiqueta al centro del objeto, en píxeles

Por defecto, las capturas de pantalla se guardan en el directorio `Imágenes\Stellarium` (Windows), en el escritorio (MacOS) o en el directorio `home` del usuario (Linux).

4. Generación del script a partir de la plantilla

El código completo del script para visualizar el caso de uso que hemos tomado como ejemplo, los fenómenos triples galileanos, está disponible en el fichero `triple_galilean.ssc` incluido en el repositorio asociado a este artículo.

Este script se ha generado a partir de la plantilla del apartado anterior mediante el código python [8] incorporado al repositorio con el nombre `stellarium_screenshots.py`. En este módulo se define la función `stellarium_screenshots()`, figura 4, que toma como argumentos precisamente los parámetros de la plantilla descritos en el apartado anterior, incluyendo además la ruta donde se almacenará el script resultante. Los parámetros relativos a la presentación de las etiquetas se presentan a la función mediante un argumento opcional de tipo diccionario, para los que, por simplicidad, se han fijado unos valores por defecto en la propia función. En el repositorio se incluye la descripción de cada argumento en formato docstring, habitual para documentar código Python.

En esta implementación, la función `stellarium_screenshots()` tiene almacenada la plantilla como variable local, y esencialmente se encarga de sustituir los parámetros de la misma gracias a la función `str.format()`, empleando un diccionario que asocia a cada parámetro su valor de sustitución. La función `str.format()` requiere que, en la plantilla, los parámetros aparezcan rodeados por llaves, `{parámetro}`: dado que las llaves son también usadas para delimitar bloques de código javascript, es necesario duplicarlas en la plantilla, es decir, `{` y `}` deben sustituirse por `{{` y `}}`, respectivamente.

Para generar el script `triple_galilean.ssc` basta con invocar esta función usando los argumentos apropiados; por ejemplo, en la figura 5 se muestra el código usado para generar el script de los fenómenos triples galileanos. Tras ejecutar este script en Stellarium, accediendo al directorio de capturas de pantalla recuperaremos las imágenes que representan los fenómenos. En la figura 6 se muestra un fragmento de la página web donde se presentan estos resultados en formato tabular [2], incorporando una columna con las imágenes generadas.

5. Algunas recomendaciones

Cuando estamos desarrollando un script para Stellarium, resulta útil mostrar mensajes indicando el contenido de ciertas variables. Para ello podemos hacer uso del método `core.debug(texto)`, donde `texto` puede construirse concatenando cadenas de texto y variables, p.ej., en el bucle podríamos incluir

```
/*
 * Configuración
 */

var lbl = LabelMgr.labelObject(                    ← Crea la etiqueta lbl, donde mostraremos la fecha
    '',                                            ← ... inicialmente, vacía;
    '{object_name}',                               ← ... asociada al objeto indicado;
    true,                                          ← ... visible;
    {size}, '{color}',                             ← ... con el tamaño de la fuente y color dados;
    '{side}', {distance}                           ← ... con la posición dada respecto del centro del objeto
);

core.setGuiVisible(false);                         ← oculta las barras de menú y herramientas
core.setMountMode('equatorial');                   ← modo ecuatorial, mantener la orientación del objeto
core.setTimeRate(0.0);                             ← pausa el tiempo
core.selectObjectByName('{object_name}');          ← selecciona un objeto a partir de su nombre
core.setSelectedObjectInfo('None');                ← ocultar la información sobreimpresa del objeto
LandscapeMgr.setFlagLandscape(false);              ← ocultar el paisaje
StelMovementMgr.setAutoMoveDuration(0.0);          ← desactivar animación al centrar el objeto
StelMovementMgr.setFlagTracking(true);             ← habilitar el seguimiento (centrar) el objeto
StelMovementMgr.zoomTo({fov_deg}, 0.0);            ← fijar FOV (campo de visión), en grados
SolarSystem.setFlagPointer(false);                 ← ocultar puntero (“punto de mira” rojo)
SolarSystem.setFlagOrbits(false);                  ← ocultar órbita del objeto


/*
 * Bucle principal
 */

var jd = [                                         ← array con las fechas julianas de los eventos
  {jd_list}
];

function getDateHM() {                             ← función auxiliar para obtener la fecha y hora local
  return core
    .getDate('local')                              ← devuelve fecha en formato yyyy-mm-ddThh:mm:ss
    .replace('T', ' ')                             ← reemplazar ‘T’ por espacio ‘ ‘
    .slice(0, -3);                                 ← dejar fuera “:ss”
}                                                  ← el formato devuelto será: yyyy-mm-dd hh:mm

function display_event(i) {                        ← función para mostrar el evento de la fecha i-ésima
  core.setJDay(jd[i]);                             ← ajustar el reloj a la fecha juliana i-ésima
  LabelMgr.setLabelText(lbl, getDateHM());         ← actualizar la etiqueta lbl con la fecha y hora local
  core.wait(0.1);                                  ← pequeña espera para que se muestre la etiqueta
}

const prefix = '{script_name}_';                   ← genera un prefijo para las capturas de pantalla
for (var i = 0; i < jd.length; i++) {              ← bucle principal, barre todas las fechas en jd,
  display_event(i);                                ← mostrando el evento en cada una de ellas,
  core.screenshot(prefix);                         ← y captura la pantalla en un fichero, usando el prefijo
}

/*
 * Limpieza
 */
LabelMgr.deleteAllLabels();                        ← elimina la etiqueta
core.setGuiVisible(true);                          ← vuelve a mostrar las barras de herramientas
```

Figura 3. *Plantilla para generar el script de capturas de pantalla.*

la línea `core.debug('Evento ' + i + ': ' + jd[i])` para mostrar el número de evento y su fecha juliana asociada en la pestaña "Log"de la consola de scripting. En este contexto, el operador ’+’ se emplea para concatenar cadenas.

En algunos casos, puede ser interesante pausar la ejecución del script hasta que el usuario decida reanudar su ejecución. Esta funcionalidad de interacción básica, prevista en el método `core.pauseScript()`, no funciona en versiones recientes de Stellarium por motivos técnicos [9]. Para suplir esta carencia, proponemos usar la función `pauseUntilPlay()`, incluida en el repositorio, que detiene la ejecución

```
def stellarium_screenshots(path,
    script_name, object_name, jds,
    fov_deg, label_fmt={}):

    stel_screenshots_tpl = """                          ← Aquí se inserta la plantilla,
       ...                                                sustituyendo { y } por {{ y }}.
    """                                                   Esta cadena incluye una indentación
                                                          que se puede eliminar con dedent()

    def_label_fmt = {                                   ← Valores por defecto para formato de
        "size": 36, "color": "#ffffff",                   la etiqueta de fecha/hora
        "side": "N", "distance": 250
    }

    script_fname = f"{script_name}.ssc"                 ← Nombre del fichero script
    jd_list = [str(jd) for jd in jds]                   ← Convertir fechas en cadenas de texto
    mapping = (                                         ← Preparación del diccionario para
        def_label_fmt                                     sustitución de valores
        | label_fmt
        | {
           "script_name": script_name,
           "jd_list": ",\n  ".join(jd_list),
           "object_name": object_name,
           "fov_deg": round(fov_deg, 4),
          }
    )

    script = (dedent(stel_screenshots_tpl)              ← Sustitución de valores sobre la
              .format(**mapping))                       plantilla (tras "des-indentarla")

    with open(Path(path) / script_fname, "w",           ← Salvar script resultante en fichero
        encoding="utf-8") as f:                           de texto, con formato UTF-8
        f.writelines(script)
```

Figura 4. *Función para generar scripts Stellarium a partir de una plantilla.*

hasta que el usuario pulsa el botón de pausa ‖ (que se alterna con ►) en la barra de herramientas inferior. Hay que tener en cuenta que para ello la barra de herramientas debe estar visible, es decir, no debemos ejecutar `core.setGuiVisible(false)`, de lo contrario no podremos pulsar el botón para reanudar el script. No obstante, si nos ocurre esto por error, podemos hacer visible la barra pulsando Ctrl+T.

6. Conclusiones

La visualización gráfica de los fenómenos que calculamos puede implicar un trabajo adicional de programación considerable, incluso partiendo de librerías de representación gráfica ya existentes.

En aquellos casos en los que los objetos involucrados estén adecuadamente representados en Stellarium, hemos mostrado cómo haciendo uso de la funcionalidad de scripting de este software, es posible generar ilustraciones que podemos incorporar en la presentación de nuestros resultados de forma sencilla. Para ello hemos desarrollado una plantilla en javascript que puede ser reutilizada en distintos proyectos, proponiendo además una implementación de referencia en python que permite, partiendo de dicha plantilla, generar los scripts para Stellarium indicando esencialmente el objeto principal y la lista de fechas julianas de los fenómenos.

Esta plantilla puede ser adaptada para generar animaciones cortas como la presentada en [2], cubriendo todo el desarrollo de un fenómeno desde principio a fin, para luego pasar al siguiente fenómeno.

```
jds = [                                          ← Fechas julianas de los fenómenos
    2460967.0585855003,
    2461386.423732,
    2463312.0232595,
    2463362.3377175,
    2463462.443047,
    2463596.925324,
    2465306.6901634997,
    2465340.340205,
    2465490.652805,
    2465641.2397775,
    2465775.727678,
    2467517.125278,
    2467986.2149755,
    2468171.1002674997,
    2468271.1420334997,
]
fov_deg = 1 / 60  # 1 arcmin                    ← Fija un campo de 1 minuto de arco

stellarium_screenshots(".",                      ← Genera el script con el nombre
     "triple_galilean",                            "triple_galilean.ssc" en el
     "Jupiter", jds, fov_deg)                      directorio actual (".")
```

Figura 5. *Uso de la función para generar el script de fenómenos triples galileanos.*

Eventos triples de satélites galileanos, 2025-2049

Inicio (UTC)	Fin (UTC)	N° eventos	Concurrentes	Satélites	Eventos	Elong. [°]	Diám. Júp. ["]	Alt. Júp. (ini)	Alt. Júp.	Alt. Júp. (fin)	Alt. Sol (ini)	Alt. Sol	Alt. Sol (fin)	Simulación (Hora local)
2025-10-18 10:34	2025-10-18 16:17	10	4	I, II, IV	[S2,T1,S4,T2]	91.2	38.9	33.3	0.8	-24.7	37.9	39.6	15.2	
2026-12-11 18:13	2026-12-12 00:50	10	4	I, II, IV	[T4,S2,S1,T2], [T4,S1,T2,T1]	112.9	40.5	-36.5	-1.5	30.1	-12.9	-59.2	-74.0	
2032-03-20 10:37	2032-03-20 14:17	8	4	I, III, IV	[S4,S3,S1,T1]	63.5	35.5	23.9	7.6	-11.0	44.5	52.8	45.5	
2032-05-09 14:30	2032-05-10 00:24	10	3	II, III, IV	[S4,S2,T3]	107.8	41.4	-47.5	-51.6	-1.3	55.3	-8.8	-34.9	
2032-08-17 19:32	2032-08-18 02:47	10	5	I, III, IV	[T3,T1,S1,S3,T4]	148.2	46.5	18.9	30.6	3.4	-4.6	-33.4	-29.9	
2032-12-30 06:18	2032-12-30 12:02	10	5	I, II, IV	[S4,T2,S2,T1,S1]	27.0	33.2	-35.8	8.6	24.6	-15.2	21.8	29.2	
2037-09-04 23:49	2037-09-05 06:42	10	4	I, II, IV	[T2,T4,S1,T1]	50.5	34.2	-19.2	30.7	56.0	-45.2	-16.8	8.0	
2037-10-08 16:59	2037-10-08 21:56	8	4	I, III, IV	[T4,S3,S1,T1]	78.4	37.2	-24.1	-30.2	-20.0	10.2	-27.2	-46.3	
2038-03-07 23:12	2038-03-08 09:17	10	3	I, III, IV	[S4,T3,T1]	122.2	42.3	55.4	3.5	-29.7	-52.5	-36.7	27.7	

Figura 6. *Ejemplo de presentación de las imágenes capturadas por el script junto con los datos de los fenómenos de interés.*

Referencias

[1] Jupiter: triple satellite phenomena, J. Meeus, Journal of the British Astronomical Association bf 112, no.5, 287 (2002).

[2] Fenómenos triples de los satélites galileanos 2025-2049, S. Díaz, https://sergiodiaz.eu/blog/20230702_galilean_triple_2049/

[3] Stellarium 24.2, G. Zotti y A. Wolf, https://www.stellarium.org

[4] ECMA-262. ECMAScript 2016 Language Specification, 7th edition, ECMA Script, June 2016, https://262.ecma-international.org/7.0/

[5] Tutorial de javascript (en español y otros idiomas), I. Kantor, https://es.javascript.info/. *Nota:* Aquellas características del lenguaje donde aparece la nota "adición reciente" en general no están soportadas en el intérprete de Stellarium.

[6] Stellarium 24.2 User Guide, G. Zotti y A. Wolf, https://www.stellarium.org/files/guide.pdf

[7] Stellarium 24.2 Scripting Engine, G. Zotti y A. Wolf, https://stellarium.org/doc/24.0/scripting.html

[8] El tutorial de Python (disponible en varios idiomas), Python Software Foundation https://docs.python.org/es/3/tutorial/index.html

[9] Github Issue 766 (2022), Stellarium https://github.com/Stellarium/stellarium/issues/766#issuecomment-1292604970