

UNIVERSIDAD DE BUENOS AIRES
COLEGIO NACIONAL DE BUENOS AIRES

UN RELOJ DE SOL DIGITAL

LUIS G. LÓPEZ SOLER

JULIO DE 2022

TÍTULO ORIGINAL: Un reloj de Sol digital
© Luis G. López Soler

Segunda edición: Julio de 2022

El presente texto fue compuesto con L^AT_EX y la clase MEMOIR.
Algunos gráficos fueron realizados con el paquete TikZ.

El código fuente completo del presente texto se encuentra en <https://github.com/lopezsolerluis/reloj-de-sol-libro>.

El código fuente completo del reloj de Sol digital se encuentra en <https://github.com/lopezsolerluis/reloj-de-sol-digital>.



Esta obra está bajo una licencia [Creative Commons Atribución-CompartirIgual 4.0 Internacional](#).



This work is licensed under a [Creative Commons Attribution-ShareAlike 4.0 International License](#).

*A Juan Valle y Ramiro Arambarri,
por permitirme abrigar la ilusión
de que las páginas que siguen
no son del todo ineficaces.*

A modo de introducción

El presente librito, o manual, es el resultado de unas lecciones escritas para un curso dictado durante el año 2021; de ahí el carácter, quizás un tanto pretencioso, de «2.^{da} edición» con que quise distinguirlo.

Dichas lecciones, a su vez, reconocen una doble inspiración. La primera de ellas llegó de la mano de uno de mis colegas docentes, Gonzalo Ciaffone, quien un día inolvidable llamó mi atención sobre el mágico objeto protagonista de estas páginas, perfectamente desconocido por mí hasta ese momento. Tras varios días dedicados a regocijarme considerando su sabia y delicada maravilla, sentí la necesidad de aplicarme a entender cómo reproducirlo: en otras palabras, a entenderlo cabalmente. Para ello tuve primero que descubrir un lenguaje de programación que me permitiera pensar y expresar la construcción del reloj de Sol digital; afortunadamente lo hallé en OpenSCAD, el cual aprendí a chapucear primero, y a ejercer luego con mayor solvencia, mientras resolvía el reloj.

La segunda inspiración de las lecciones es más fácil de declarar, aun cuando resulta mucho más misteriosa: soy docente, y siento la invencible necesidad de enseñar todo lo que aprendo.

El formato dialogado y con una cierta aspiración de novela

que elegí para el desarrollo de las lecciones y que mantuve para este manual me pareció apropiado y divertido cuando urdí su inicio; con el correr de las páginas comprobé luego que también me ayudaba a llevar adelante la propia exposición del contenido teórico. No pocas fueron las encrucijadas en las cuales el diálogo entre las protagonistas fue el que me permitió descubrir la única manera de avanzar a través de asuntos necesariamente difíciles de exponer. Espero que este formato resulte también grato para el lector.

Otra virtud surgida de la escritura de las lecciones pude comprobarla en el código producido durante la misma: la versión que obtuve mientras las preparaba resultó mucho mejor que la que me permitió imprimir mi primer reloj de Sol digital. Comparando ambos códigos no puedo atribuir la diferencia exclusivamente a una mejor comprensión y dominio del lenguaje de OpenSCAD: no lo había ejercido tanto entre uno y otro. Tengo para mí que fue el diálogo entre Cecilia y Antonia —las mentadas protagonistas del librito— el que me permitió reflexionar más y mejor acerca del objeto que estaba (o estábamos ya) escribiendo, lo cual resultó en un texto saludablemente más terso, claro y optimizado. Esta virtud no debiera tomarme por sorpresa, después de todo: de manera muy clara puede leerse en uno de los consejos que enriquecen el libro *The Pragmatic Programmer*.¹ En el tema 20 —Debugging (corrección de errores)— los autores proponen un proceder que denominan «del patito de hule», y que consiste esencialmente en obligarse a explicar a alguien —o, incluso, a *algo*— lo que se quiere resolver.

Quizá pueda parecer extraño a quien no escribe regularmente que el solo hecho de poner en palabras una cuestión ayude a entenderla mejor, o incluso a resolvlerla; sin embargo, es una realidad

¹*The Pragmatic Programmer: Your Journey to Mastery, 20th Anniversary Edition*, Andrew Hunt y David Hurst Thomas, 2019.

conocida desde hace tiempo. Se dice que el propio Montaigne, en el siglo XVI, ya había afirmado que nada aclara tanto las ideas como ponerse a escribirlas.²

Atribuí al principio de esta introducción el origen inmediato de estas páginas a un curso dictado durante el año 2021. No sería honesto de mi parte dejar de confesar que su éxito resultó, cuando menos, discutible: de los 15 participantes que se sintieron convocados por la propuesta sólo dos me acompañaron hasta el final. Esta abrumadora deserción seguramente se debe a una suma de causas variadas, aun cuando la mayoría de ellas, ¡ay de mí!, sin duda deben reconocerme como culpable. En cualquier caso me gustaría ofrecer en mi defensa la frase de Séneca que uno de los dos alumnos que decidió terminar el curso eligió para adornar su reloj de Sol digital, y que yo consideré luego apropiado estampar al frente de los capítulos de esta segunda edición: «*Non est ad astra mollis e terris via*»: No hay un camino fácil de la tierra a las estrellas. Serás tú, querido e improbable lector, quien decida si esta inscripción es una bienvenida o una amenaza.

Por último, me siento también en la obligación de sumarme al consejo universal: si alguien pretende aprender a programar, no deberá limitarse a leer, sino lanzarse a escribir. El aprendizaje de cualquier idioma exige su ejercicio: los lenguajes de programación no se encuentran al margen de esta necesidad. Es por esto por lo que te aconsejo que escribas todos los textos que tu curiosidad te sugiera a medida que avanzas a través de los capítulos, sin miedo frente a la pantalla en blanco ni a los errores: El único *bug* incorregible es el que no se escribe.

²No pude encontrar la referencia precisa; la noticia me llega de Paul Graham quien en su libro *On Lisp* (Prentice Hall, 1993) la trae en la página 2.

CONVENCIONES

NÚMEROS DE LÍNEA

A fin de facilitar las referencias dentro del texto, se añaden números de línea al código informático en su margen izquierdo:

```
1 $fn = 200;  
2  
3 sphere(r=10);  
4 cube([25,10,20]);  
5 cylinder(h=15, r=5);
```

Por otra parte, cuando el código consta de una sola línea, tal numeración se omite:

```
cube([20,30,40]);
```

PRIORIDAD DE LOS OPERADORES ARITMÉTICOS

OpenSCAD respeta la prioridad usual de los operadores aritméticos; de esta forma, una expresión como

```
1+2*3-4
```

se interpreta como si estuviera escrita así:

```
1+(2*3)-4
```

y, por lo tanto, devuelve “3” como resultado.

Non est ad astra mollis e terris via.

Hercules furens

LUCIUS ANNAEUS SENECA

— 1 —

Cecilia Payne, PhD.

CECILIA PAYNE SE sentía feliz: ya era PhD en Astronomía. A pesar de las interrupciones de Antonia y de los *bugs* del AN^IE¹ fue capaz de conquistar, en una reflexiva epifanía, la significación profunda de las diferencias espectrales entre las estrellas, gracias a lo cual pudo calcular las proporciones relativas de sus elementos químicos constituyentes. El título de su trabajo era suficientemente modesto: «*Atmósferas estelares: una contribución al estudio observational de la alta temperatura en las capas inversoras de las estrellas*»², pero ya sentía que, en no muchos años, sería saludado como el trabajo singular más importante de la historia de la Astronomía.

Cecilia respiró hondamente y, tras un largo suspiro, recordó sus conversaciones con Fumington —director del Observatorio de Harvard— durante las últimas etapas de la elaboración de su monografía. «¡Conversaciones..!» —pensó— «Más bien monólogos...». Monólogos en los cuales Fumington la convenció finalmente de mitigar sus resultados, ciertamente revolucionarios, con los cuales demostraba que las estrellas estaban compuestas mayormente por

¹Para mayor referencia, véase el manual del programa AN^IE («ANálisis Numérico de la Información Espectral»).

²Payne, C. H. (1925). *Stellar Atmospheres; a Contribution to the Observational Study of High Temperature in the Reversing Layers of Stars* (PhD Thesis). Radcliffe College.

Hidrógeno. A Fumington, como al resto de los sabios de esos años, le parecía que la composición química del Sol *debía* ser igual a la terrestre, y por eso Cecilia *debía* morigerar sus resultados... Trató de recordar en qué momento preciso la había convencido; por supuesto, no lo logró, y nunca lo lograría. Así eran los monólogos de Fumington: largas retahílas sintácticas de palabras con una apariencia superficial de lógica, que luego en el recuerdo —incluso inmediato— se difuminaban en una nube de humo. «¿Cómo fue que le dije que sí?» —se preguntó Cecilia, acaso más retóricamente que buscando una precisión que, sabía, no podría lograr en un asunto en el que mediara Fumington. Pero ya no importaba: en breve su tesis en todo su contenido original resurgiría y sus fieles resultados serían aplaudidos por generaciones de astrónomos, profesionales y aficionados, sin posibilidad alguna de quedar ya sepultados bajo una insoportable montaña de humo denso.

Cecilia volvió a suspirar hondamente. Estaba sentada en un amplio banco de los jardines de Harvard. El otoño apenas comenzaba y la suave luz de un fresco mediodía se colaba entre los huecos de las hojas del aoso y vital árbol a cuya sombra el banco estaba. Elevó su rostro y recibió en sus párpados cerrados la delicada luz solar filtrada entre las hojas, que sentía apenas como un brillo cálido acariciándola. Y mientras pensaba que por fin sabía de qué estaba hecho el Sol, se sorprendió una vez más al descubrir la dicha que ese conocimiento le deparaba. Trató de distinguir, en ella, alguna nota de vanidad u orgullo; pero no: con perplejidad y gratitud comprobó que el hecho mismo de conocer, de saber, la hacía feliz. Siempre había sido una curiosa, una inquieta: ahora podía comprobar que, sencillamente, lo seguía siendo pero con el nombre de astrónoma profesional. Seguía siendo aquella misma chiquilla que sentía una irresistible atracción por los misterios del mundo y de la fantasía: la clara luz del Sol vibrando en el agua durante el día y los oscuros espectros de Poe en la profunda noche de su cuarto infantil; los íntimos secretos de la luz de las estrellas

palpitando para ella en los espectros de absorción en su adulta oficina.

Se atrevió a abrir los ojos. Los huecos entre las hojas eran tan pequeños que ningún rayo de Sol alcanzaba para cegarla. Antes bien, parecían puntitos de luz en un fondo vegetal: estrellas titilando en una noche verde oscuro. Bajó sus ojos al suelo, y vio la luz del Sol en multitud de manchas moviéndose en el piso, en el banco, en sus manos. Cecilia miraba ensoñadoramente el movimiento armónico y delicado de esas manchas y sintió que, si bien era feliz, también una sombra comenzaba a pesar en su alma: ahora que había terminado la tesis que había colmado su tiempo y su energía los últimos años, ¿qué? ¿Qué seguía ahora? Una sensación de vacío había comenzado, lentamente, a inquietarla durante los días anteriores.

Las manchas a sus pies, indiferentes a sus cavilaciones, seguían su movimiento caprichoso. Cecilia casi ya no reparaba en ellas, hasta que un conjunto de las mismas llamó imperceptiblemente su atención. Parecían moverse juntas. No sólo eso: parecían formar una suerte de patrón. Cecilia se frotó los ojos: parecían dispuestas según un arreglo matricial. Su sorpresa se convirtió en alarma al comprobar que esas manchas formaban, con una claridad mágica, un símbolo numérico.

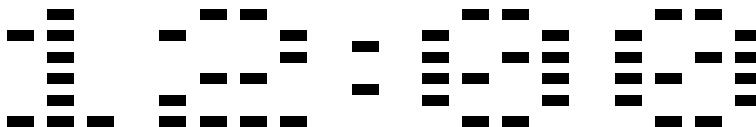


Figura 1.1: Las misteriosas manchas de luz solar que alarmaron a Cecilia en el suelo de los jardines de Harvard un mediodía otoñal.

Se irguió en el banco; sus ojos se clavaron, tan abiertos e inquietos como su curiosidad, en esas manchas de luz ordenadas y demenciales. Cecilia era tan joven que ni se le ocurrió la posibili-

dad de haberse vuelto loca: su cabeza se hizo un súbito remolino en busca de una explicación racional. Instintivamente echó su cabeza hacia atrás, elevando sus ojos al cielo, y pudo apreciar con sorpresa que, de una ventana en el primer piso del edificio que se encontraba a su espalda se asomaba temerariamente un cuerpo que sostenía en sus manos un pequeño y alargado objeto, del cual parecían salir rayos de luz. Entrecerró los ojos tratando de reconocer a la persona asomada cuando recordó súbitamente a quien pertenecía la oficina cuya ventana enmarcaba ese cuerpo.

—¡Cecilia! —la voz de Antonia Maury resonó, clara y socarrona, en los jardines de Harvard—. ¿Qué hacés ahí sentada, con cara de pavota? Vení, subí que te muestro lo que acaba de inventar mi inteligencia superior.

Cecilia sintió que una sonrisa casi le rompía la cara de lado a lado. Rápida y feliz se levantó, y decididamente se dirigió a la puerta que la conduciría, a través de los interminables pasillos de Harvard, a la oficina de Antonia y a una nueva aventura.

Antonia Maury, Aydte.

CUANDO CECILIA entró en la oficina de Antonia la encontró sentada frente a la computadora. En sus manos se veía, ahora con claridad al encontrarse más cerca y sin ser deslumbrada por la luz del Sol, el objeto semicilíndrico con el que seguramente creó las manchas que tanto la alarmaron en el jardín. Tendría unos 20 centímetros de longitud y un diámetro de aproximadamente 5 centímetros. Presentaba curiosos agujeros o ranuras, dispuestos en lo que parecía una especie de azar controlado por una ley secreta.

—Como recordarás, hace unos días la familia Ciaffone Hallak, patrocinante de esta benemérita institución, donó una impresora 3D a Harvard —rememoró Antonia con voz clara—. Fumington quería que diseñara unos posavasos para la sala de reuniones —Antonia subrayó esta última palabra con la voz y con el gesto, y Cecilia no pudo reprimir una risita cómplice recordando un suceso no muy lejano—. Pero a mí se me ocurrió algo más... interesante.

Cecilia sonreía con delicada ternura mientras escuchaba y miraba a Antonia. Antonia se mantenía siempre igual: periférica, brusca, irritable, indiferente, vulnerable. Siempre en guerra con Fumington y ansiosa por compartir sus resultados —de los cuales era la primera en aburrirse y dejar de lado— con sus compañeras, sin importarle lo que éstas estuvieran haciendo al momento de

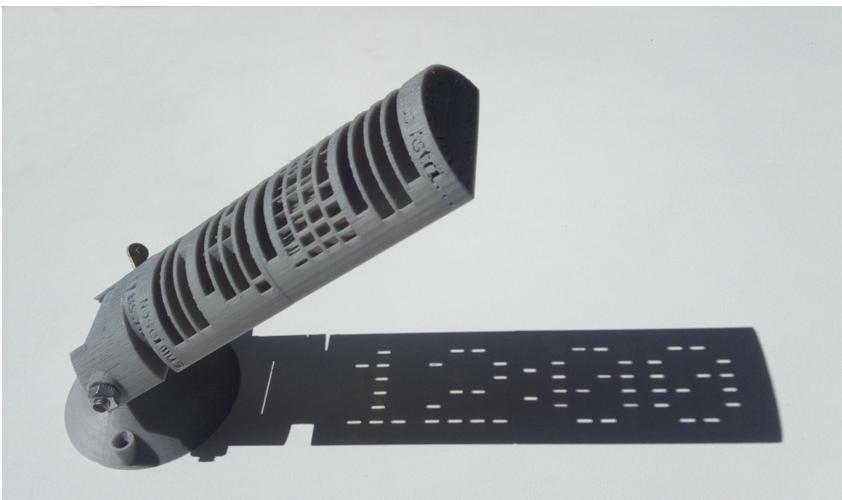


Figura 2.1: El mágico objeto con el que Antonia produjo las manchas que alarmaron a Cecilia, y que pronto encenderá su entusiasmo.

interrumpirlas con los productos de su impredecible pulsión por hacer y deshacer. Cecilia había descubierto que el tiempo, junto a ella, pasaba mejor.

— 3 —

La idea

—¿Vos viste los relojes de Sol, ¿no? —empezó Antonia, y Cecilia asintió levemente, resignada ya hacía tiempo a que le hiciera preguntas que suponían en ella ignorancia acerca de los asuntos más elementales de la Astronomía—. Pues bien —prosiguió—, diseñé uno que aprovecha el mismo principio, pero en lugar de indicar la hora ‘analógicamente’ mediante la sombra de un gnomón, la señala dejando pasar la luz por huecos dispuestos de manera tal que sólo los atraviese limpiamente cuando el Sol se encuentra a una altura determinada sobre el horizonte.

Antonia, una vez más, había caído en uno de sus defectos favoritos: priorizar la fluidez sintáctica y elegante brevedad superficial de un período antes que la claridad y expresividad de su contenido. Cecilia arqueó las cejas, esperando que, al dejarla hablar, lentamente fuera quedando más o menos en claro su idea.

—La idea no es mía —continuó Antonia—; Kenneth Falconer demostró matemáticamente en 1987 que existe un objeto fractal cuyas sombras, si es iluminado desde distintas direcciones cualesquiera, responde a un diseño previo arbitrario.¹ O sea, si elijo un conjunto arbitrario de direcciones y sombras vinculadas uno

¹Kenneth Falconer, *Fractal Geometry: Mathematical Foundations and Applications*, 2nd Edition (2003), John Wiley & Sons Ltd.

3. LA IDEA

a uno, existe siempre un objeto fractal que garantiza la formación de cada sombra al ser iluminado desde la dirección que le corresponde.

A Cecilia le pareció que empezaba a comprender. Lo sintió en todo su cuerpo; siempre le pasaba lo mismo: las ideas nuevas, raras e inesperadas le producían una felicidad e inquietud físicas. En esos momentos se alegraba de haber elegido la Ciencia como profesión.

—Este coso —de pronto Antonia abandonaba su pretensiosa elegancia verbal para caer en un vocabulario preescolar, mientras le pasaba a Cecilia el objeto misterioso a fin de que lo pudiera examinar más de cerca— puede considerarse el reverso del objeto fractal de Falconer: en lugar de expresarse con sombras, lo hace con luz. Bueno, por supuesto, tampoco es un fractal, ni deja pasar la luz de manera distinta a cada instante. Supongo que tal objeto sería imposible de realizar en la práctica. En todo caso, miralo y decime si no es hermoso.

Cecilia pensó que Antonia tenía razón. El objeto que tenía entre sus manos y que volvía y revolvía en ellas era muy hermoso; tanto más que aún no lo comprendía del todo y por lo tanto estaba cargado con todas las promesas del misterio y las posibilidades de la geometría. «Este objeto» —pensó Cecilia— «será infinito mientras no lo comprenda del todo». No obstante, sabía que la pulsión por entender la vencería, y la finitud se cerniría sobre ese semicilindro de plástico que sus dedos acariciaban.

—¿No me vas a preguntar cómo lo hice? —Antonia hizo pucherito, simulando contrariedad y sonriendo con los ojos.

Cecilia salió con una sonrisa de su ensimismamiento y se dispuso a escuchar.²

² Antonia parece olvidar una inspiración más cercana en el tiempo, y más concreta: en el sitio <https://www.thingiverse.com/thing:1068443> se muestra y comparte un modelo que acicateara al autor de estas páginas para replicar desde cero, lo mejor posible, tan mágico objeto. (Nota del Editor)

— 4 —

OpenSCAD

—EN UN PRINCIPIO lo intenté con FreeCAD¹ —confesó Antonia—. Pero no tardé mucho en comprender que no era la herramienta adecuada.

—¿Probaste entonces con el Fusion3D, o con el SolidWorks? —preguntó ingenuamente Cecilia, y al ver la cara de Antonia se arrepintió inmediatamente.

—¿Qué?? ¿Programas propietarios??? —replicó su amiga, indignada. Cecilia pudo ver que iba a seguir en el mismo tono, pero con sorpresa apreció que se contuvo. Antonia cerró los ojos y, haciendo un esfuerzo, suspiró—. En fin, disculpame —susurró—. Supongo que también son buenos programas, pero ya sabés que no me siento cómoda más que con programas de código fuente abierto. Sí, es un capricho, lo sé —admitió—; pero, ¿qué querés...? —y Antonia completó la idea encogiéndose de hombros.

Cecilia, con una sonrisa, la invitó a continuar.

—Entonces fue que encontré OpenSCAD² —el tono de Antonia recobró su ímpetu natural—. No sabés; está buenísimo. No creás objetos 3D con el ratón, ni con botones de un menú: los creás programando, escribiendo. Todo lo creás con texto; el ratón apenas

¹<https://www.freecadweb.org/>

²<https://www.openscad.org/>

4. OPENSCAD

lo usás para rotar o desplazar la escena que vas creando.

Se notaba en el tono vibrante de la voz de Antonia que estaba commovida. A Cecilia le costaba imaginar porqué alguien podía verse motivado por lo que en última instancia no era otra cosa que una limitación, tal como es la imposibilidad de usar el ratón, los botones y los menús. Pero hasta en la locura de Antonia había una lógica; tal vez retorcida, tal vez excéntrica, pero una lógica al fin. Y, por otra parte, el mágico objeto que guiaba la luz del Sol seguía ahí, en sus manos: dócil, mudo y todavía secreto. Así que la dejó continuar.



Figura 4.1: Pantalla de bienvenida de OpenSCAD.

—Cuando abrís OpenSCAD aparece una ventana no muy distinta a la de tantos otros programas —adelantó Antonia—. En ella se te ofrece la opción de comenzar con un texto nuevo, abrir uno ya escrito por vos u otra persona, o incluso chusmear entre una gran cantidad de ejemplos.

»Si elegís empezar un texto nuevo aparecerá una ventana dividida en otras cuatro —agregó Antonia, señalando el monitor de la computadora con el mentón, que lucía como en la figura 4.2—. En la izquierda (1) escribís tu texto, en la central (2) se ve

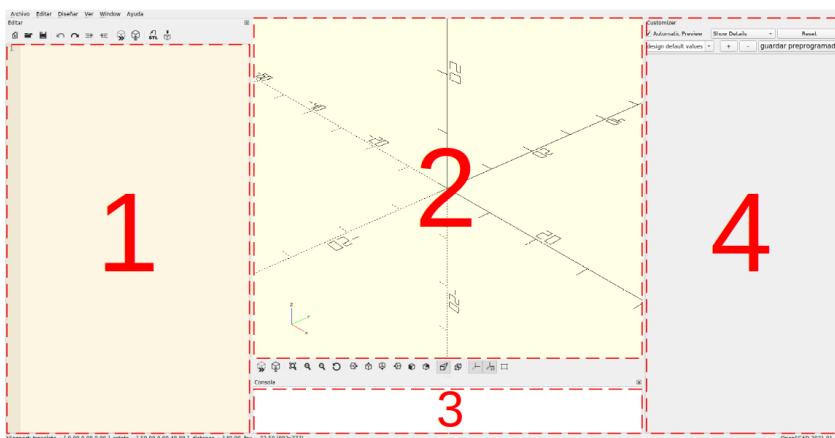


Figura 4.2: Área de trabajo de OpenSCAD.

el resultado del mismo y en la de abajo (3) aparecen mensajes ocasionales, que pueden sancionar errores o confirmar el éxito. Te aconsejo que cierres la de la derecha (4) pulsando sobre la pequeña 'x' de su esquina superior derecha: su uso puede ser de utilidad, pero sólo más adelante.

Cecilia no pudo evitar sentir una suerte de delicioso vértigo ante la página en blanco que el monitor lanzaba a su rostro.

—¿Qué onda los ejemplos que trae OpenSCAD? —preguntó.

—Están buenísimos. Pero espero que no los entiendas para nada —contestó Antonia, con un gesto malicioso.

—¡Hey! ¿Por qué? —preguntó Cecilia, visiblemente asombrada.

—Porque si los entendés por tu cuenta, ¿a quién le voy a contar cómo hice el reloj solar? —remató Antonia sonriendo con los ojos y haciendo, una vez más, pucherito.

— 5 —

Creación de objetos

—MIRÁ —empezó Antonia—; para crear, no sé, un cubo, escribís esto:

```
cube ([10 ,20 ,30]);
```

»Acto seguido, pulsás la tecla  para visualizar el resultado de tan expresiva sentencia —añadió con un cierto aire pedagógico.

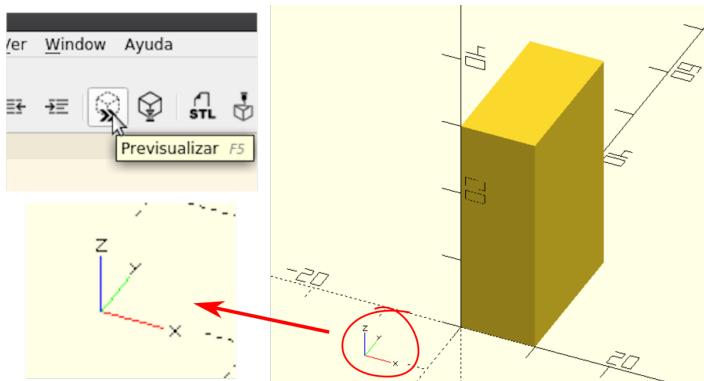


Figura 5.1: Antonia crea un `cube`.

Cecilia pensó que el objeto recién creado por Antonia no era precisamente un cubo, pero pudo captar la idea: los valores entre

5. CREACIÓN DE OBJETOS

corchetes indicaban el tamaño de un paralelepípedo recto de base rectangular en cada uno de los tres ejes cartesianos. En otras palabras, el objeto medía 10 en X, 20 en Y y 30 en Z. La identidad de las coordenadas pudo deducirla gracias a los pequeños ejes cartesianos que se veían abajo a la izquierda en la figura 5.1. Por lo demás, decidió ser indulgente con el nombre del objeto: supuso que a ningún programador le gustaría escribir *paralelepípedo recto de base rectangular* (aun en inglés) un número de veces mayor que dos.¹

Antonia continuó:

—Para un cilindro escribís:

```
cylinder(h=30 , r=10);
```

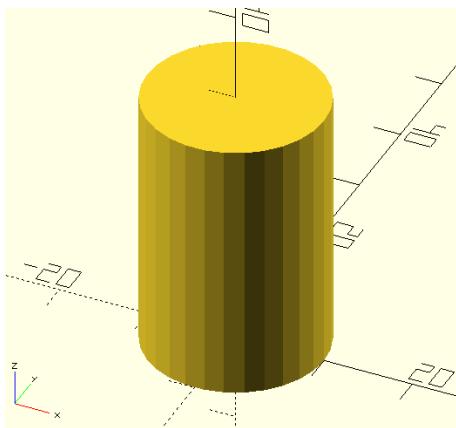


Figura 5.2: Antonia crea un `cylinder`.

A Cecilia no le costó mucho desentrañar la lógica de esa línea: un cilindro parecía definirse por su altura ($h=30$) y su radio ($r=10$).

—Antonia —preguntó Cecilia—, ¿en qué unidades mide las distancias OpenSCAD?

—Milímetros —contestó Antonia.

¹Cecilia parece ignorar que en español existe una palabra suficientemente breve que designa dicho objeto geométrico: *ortoedro*. (Nota del Editor)

—Lógico —concedió Cecilia—. Otra cosa: ¿Por qué el supuesto cilindro no tiene un contorno debidamente circular, sino facetado?

—Es una buena pregunta —comenzó Antonia, y Cecilia supo entonces que iba a tener que fumarse un poco de humo: Antonia sólo consideraba buenas aquellas preguntas para las cuales no contaba con una respuesta satisfactoria—. El formato de los objetos tridimensionales que OpenSCAD produce en última instancia —agregó—, a fin de ser luego impresos en 3D, sólo permite acotarlos con superficies planas: triángulos o rectángulos. De esa manera, una superficie curva debe ser reducida a un cierto número de caras planas.

Cecilia intentó negociar:

—Entiendo que pueda ser un tanto ingenuo pretender objetos euclídeamente cilíndricos, pero ¿no se podría suavizar un poco la superficie de ese tubo, al menos a la vista y luego al tacto?

—Por supuesto —Antonia sonrió—; la variable de sistema `$fn` define la cantidad de caras que una superficie curva debe tener; mirá —agregó, escribiendo rápidamente el ejemplo de la figura 5.3.

»Creo que se me fue la mano con 5000 —confesó Antonia con una ligera risita—, pero la idea es ésa: con `$fn=5000` le decís a OpenSCAD que los objetos deben dividir sus partes curvas en 5000 caras.² En fin, ya debés sospechar cómo se hace una esfera:

»Creo que es un buen momento para pasar algunas ideas básicas en limpio —Antonia retomó su tono didáctico—. Cada una de las líneas que escribí hasta ahora puede contemplarse como una instrucción o sentencia; una suerte de ‘orden’ o ‘indicación’ a OpenSCAD con un sentido propio. Algunas sirven para crear un objeto (como `cube([10,20,30])`), mientras que otras advierten acerca de cómo deben concretarse dichos objetos (como `$fn=200`).

²Los autores de OpenSCAD sugieren no emplear valores mayores a 100 para `$fn`. (Nota del Editor)

5. CREACIÓN DE OBJETOS

```
1 $fn=5000;  
2  
3 cylinder(h=30,r=10);
```

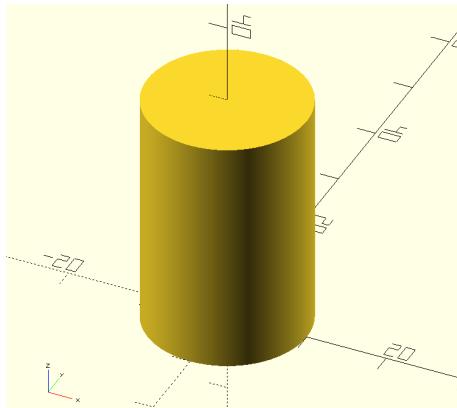


Figura 5.3: Antonia crea un `cylinder` con un tanto exagerado pero indiscutiblemente eficaz `$fn=5000`.

```
1 $fn=200;  
2  
3 sphere(r=10);
```

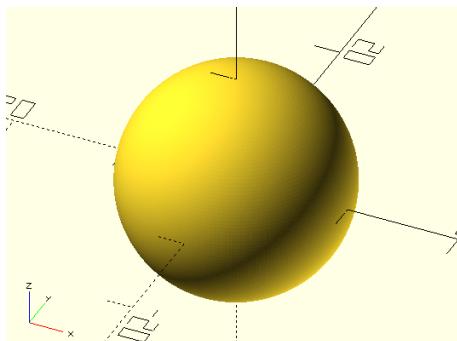


Figura 5.4: Antonia crea una `sphere`.

»Las sentencias finalizan con un punto y coma (;). Las líneas en blanco no tienen otro efecto que organizar mejor, a la vista, el texto: podés usarlas a discreción.

»¡Ah! ¡Casi me olvidaba! —soltó con vivacidad Antonia, y Cecilia no pudo distinguir si estaba o no sobrereactuando el tono de alarma—. Nunca te olvides de grabar tu trabajo: la combinación

 +  debe ser una de tus mejores aliadas.



Figura 5.5: Icono para guardar el texto.

Cecilia no pudo evitar un leve bostezo.

— 6 —

Traslación de objetos

CECILIA DECIDIÓ que era momento de ponerse a escribir:

```
1 $fn = 200;  
2  
3 sphere(r=10);  
4 cube([25,10,20]);  
5 cylinder(h=15,r=5);
```

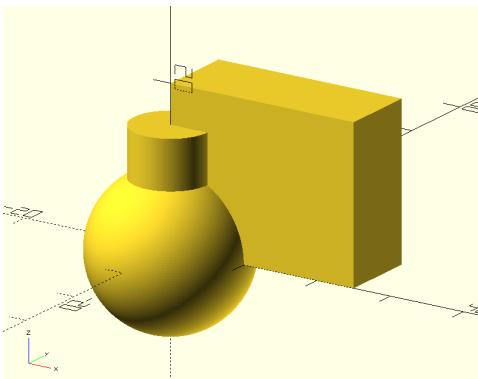


Figura 6.1: Cecilia escribe sus primeros objetos.

—Lindo —dijo Antonia displicentemente, y Cecilia sintió que sus mejillas levantaban temperatura: le molestaba sobremanera la sola posibilidad de que se burlaran de ella.

—Estoy probando, ¿ok? —dijo con tono seco, y agregó—: ¿Serías tan amable de indicarme cómo hago para desplazar los objetos, a fin de que no se solapen o que, en todo caso, lo hagan

donde yo quiera?

6.1. CENTER=TRUE

—Sí, por supuesto —Antonia pareció no notar el disgusto de Cecilia; esa curiosa insensibilidad no era rara en ella, por otro lado—. En primer lugar, habrás notado que los cubos¹ surgen con uno de sus vértices coincidiendo con el origen de coordenadas:

```
cube([25,10,20]);
```

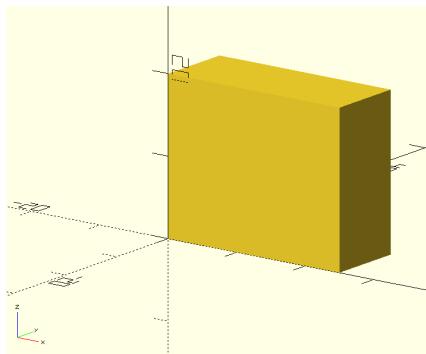


Figura 6.2: Un `cube` es creado con un vértice en el origen de coordenadas...

»Si querés que aparezcan centrados en dicho origen debés agregar la indicación `center=true` —explicó, mientras escribía el ejemplo de la figura 6.3.

»Las esferas, por su parte, aparecen centradas en el origen de coordenadas de por sí. En el caso de los cilindros, en principio son creados con su eje centrado en el eje Z y su base apoyada en el plano XY —dijo, demostrándolo en la figura 6.4.

¹Recordamos al lector que un término más ajustado sería *ortoedro*, pero en fin. (Nota del Editor)

```
cube([25,10,20],center=true);
```

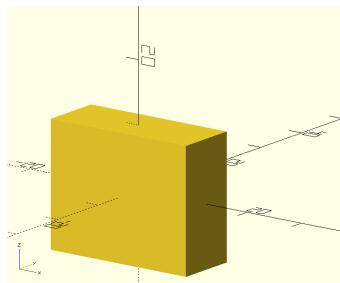


Figura 6.3: ...salvo que reciba la indicación suplementaria `center=true`.

```
1 $fn = 200;
2
3 cylinder(h=15,r=5);
```

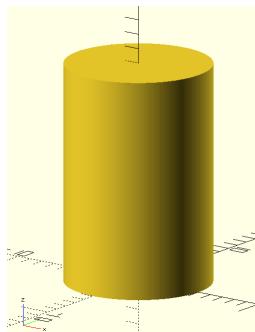


Figura 6.4: Un `cylinder` es creado apoyado sobre el plano XY...

»Por otra parte, el añadido `center=true` provoca que *todo* el cuerpo del cilindro resulte centrado con respecto al origen, como podés apreciar en la figura 6.5.

6.2. TRANSLATE

»Ahora bien, para trasladar un objeto de la manera más general posible debés precederlo por la indicación `translate` —anunció Antonia, escribiendo el ejemplo de la figura 6.6.

6. TRASLACIÓN DE OBJETOS

```
1 $fn = 200;  
2  
3 cylinder(h=15, r=5,  
            center=true);
```

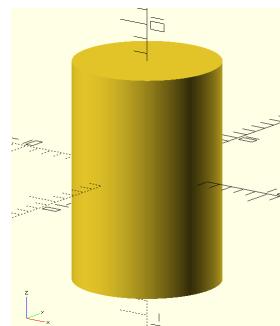


Figura 6.5: ...mas con la indicación `center=true` resulta centrado en el origen de coordenadas.

```
1 $fn = 200;  
2  
3 sphere(r=10);  
4  
5 translate([15, -5, -10])  
6   cube([25, 10, 20]);  
7  
8 translate([-20, 0, -7.5])  
9   cylinder(h=15, r=5);
```

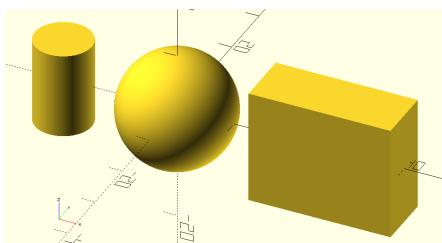


Figura 6.6: La transformación `translate` permite colocar un objeto donde al escritor le plazca.

Cecilia se acercó involuntariamente a la pantalla, entrecerrando ligeramente los ojos mientras analizaba lo escrito por Antonia. La instrucción `translate` parecía bastante elocuente: trasladaba el objeto al cual se aplicaba a continuación una cierta distancia en X, Y y Z. En el caso particular del cubo reciente lo movía 15mm en X, por ejemplo. Le pareció simpático el empleo de números negativos: mover el cubo -10mm en Z equivalía a hacerlo descender.

—Fijate que la línea `translate` no termina con un punto y coma

—Antonia parecía sentir a veces la necesidad de interrumpir los pensamientos de Cecilia—; eso se debe a que no es una instrucción en sí misma, sino que es una suerte de transformación que se aplica al objeto inmediato siguiente.

—Suena lógico —dijo ésta tras considerarlo unos instantes, y preguntó:

—¿Son necesarios los dos espacios que preceden a `cube` y `cylinder` en el texto?

—No —respondió Antonia—; son otra mera comodidad visual. Podés omitirlos si querés. De hecho, podés escribir

```
translate([15,-5,-10]) cube([25,10,20]);
```

en una misma línea y funciona igual. Con el tiempo, y a medida que escribas, vas a encontrar la mejor disposición visual para tus textos; lo importante es que resulten visualmente claros. La claridad en este género literario, no sólo en cuanto al contenido semántico sino en el aspecto meramente visible, es fundamental: vas a descubrir que tus propios textos pueden resultarte extraños tras unos días de no frecuentarlos. En esos momentos agradecerás haberlos escrito de la manera más clara posible.

Cecilia sintió que ahora podía escribir algo bastante más bonito. Después de un buen rato consiguió enhebrar unas cuantas esferas:

```
1 $fn = 200;
2
3 sphere(r=16);
4 translate([0,0,24])
5   sphere(r=8);
6 translate([0,0,36])
7   sphere(r=4);
8 translate([0,0,42])
9   sphere(r=2);
10 translate([0,0,45])
11   sphere(r=1);
12 translate([0,0,46.5])
```

6. TRASLACIÓN DE OBJETOS

```
13     sphere(r=.5);  
14     translate([0,0,47.25])  
15     sphere(r=.25);
```

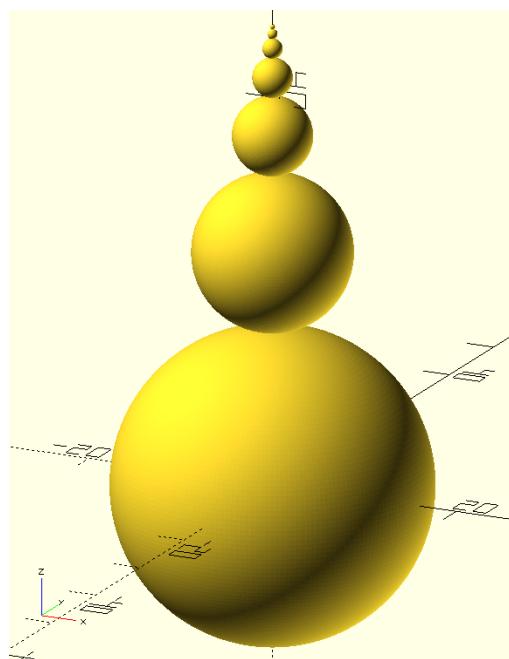


Figura 6.7: Cecilia se anima y apila una serie de esferas.

—No está mal... —aprobó Antonia, admirando con una amplia sonrisa la figura 6.7.

Cecilia estuvo de acuerdo, y siguió jugando:

```
1 $fn = 200;  
2  
3     sphere(r=16);  
4     translate([0,0,24])  
5     sphere(r=8);  
6     translate([0,0,36])
```

```
7     sphere(r=4);
8     translate([0,0,42])
9     sphere(r=2);
10    translate([0,0,45])
11    sphere(r=1);
12    translate([0,0,46.5])
13    sphere(r=.5);
14    translate([0,0,47.25])
15    sphere(r=.25);

16
17    translate([60,0,0])
18    cube([32,32,32],center=true);
19    translate([60,0,24])
20    cube([16,16,16],center=true);
21    translate([60,0,36])
22    cube([8,8,8],center=true);
23    translate([60,0,42])
24    cube([4,4,4],center=true);
25    translate([60,0,45])
26    cube([2,2,2],center=true);
27    translate([60,0,46.5])
28    cube([1,1,1],center=true);
29    translate([60,0,47.25])
30    cube([0.5,0.5,0.5],center=true);
```

—¡Extraordinario! —exclamó Antonia, y en sus ojos bailaba un fulgor especial mientras contemplaba alternativamente el texto de Cecilia y la figura 6.8—. Este texto me encanta, porque nos va a permitir reflexionar un poco sobre el problema particular que atrapó tu atención.

—¿Qué problema particular? —preguntó Cecilia con cándida sinceridad—. Sólo quise poner en práctica los poquitos elementos que vimos hasta ahora.

—En principio, es posible que así haya sido —replicó Antonia fingiendo paciencia—. Pero podrías haberlo hecho de mil modos distintos. Pensemos juntas *por qué* elegiste éste y no otro. ¡Ojo!

6. TRASLACIÓN DE OBJETOS

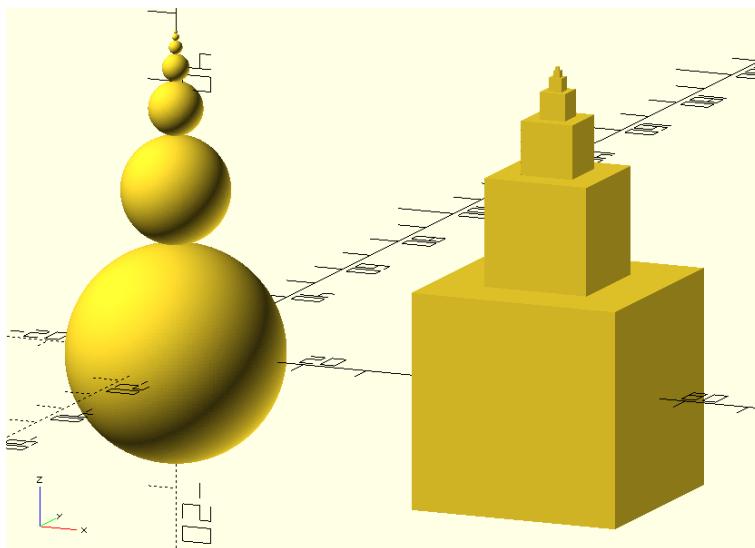


Figura 6.8: Esferas y cubos apilados por Cecilia.

—se atajó—; no estoy pensando en hacerte psicoanálisis a la violeta; si querés, digamos que vamos a analizar el texto a ver qué nos cuenta.

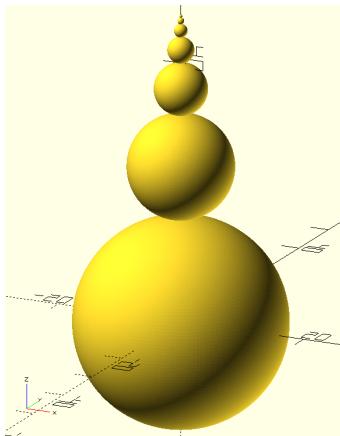
Cecilia, incluso acostumbrada a las excentricidades de Antonia, no pudo evitar cierto recelo.

— 7 —

Análisis literario – I

ANTONIA TOMÓ el control del teclado. Sus ojos relucían con un brillo que no podía explicarse exclusivamente por el reflejo de la luz del monitor. Sus labios se movían levemente, como si estuviera pronunciando internamente las líneas que Cecilia acababa de escribir.

```
1 $fn = 200;  
2 sphere(r=16);  
3 translate([0,0,24])  
4   sphere(r=8);  
5 translate([0,0,36])  
6   sphere(r=4);  
7 translate([0,0,42])  
8   sphere(r=2);  
9 translate([0,0,45])  
10  sphere(r=1);  
11 translate([0,0,46.5])  
12  sphere(r=.5);  
13 translate([0,0,47.25])  
14  sphere(r=.25);
```



7.1. EXPRESIONES MATEMÁTICAS

—Empecemos por las alturas: ¿Cómo elegiste los valores 24, 36, 42, 45, etc., en la coordenada Z de los [translates](#)? —preguntó Antonia con una sonrisa.

Cecilia miró atentamente a su amiga; por un momento cruzó por su mente la posibilidad de que estuviera jugando con ella. Pero aun así decidió contestar:

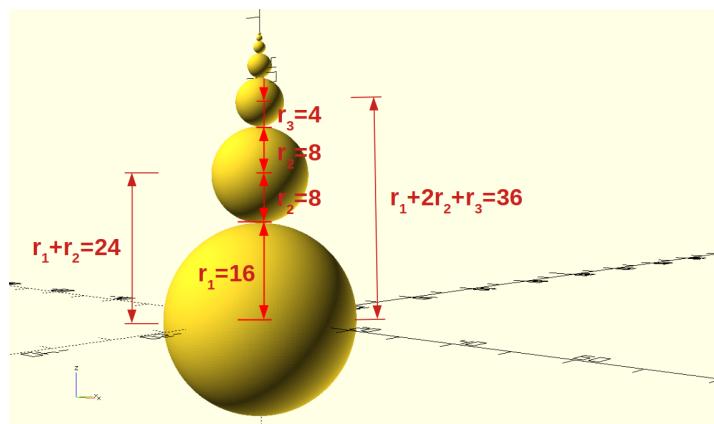


Figura 7.1: Cálculos explícitos hechos por Cecilia para apilar esferas de radios decrecientes.

—La primera esfera, centrada en el origen, es de radio 16. La segunda, de radio 8, quería que fuera tangente a la primera. Luego, la altura de su centro debería ser igual a la suma de ambos radios: 24. La tercera esfera era de radio 4: por lo tanto, la altura de su centro (a fin de resultar tangente a la segunda) debía igualar la suma de su radio, el radio de la primera y dos veces el radio de la segunda. El razonamiento se repite para las demás.

Cecilia, tras concluir su explicación, en la que empleó un tono deliberadamente lento, volvió a mirar atentamente a Antonia a fin de comprobar en su gesto si había habido alguna intención

juguetona en su pregunta inicial. Pero como su amiga mantenía la misma sonrisa franca que manifestaba cuando un asunto despertaba su interés, Cecilia decidió abandonarse a sus preguntas, ya sin temores.

—Entiendo la lógica, pero ¿por qué hiciste las cuentas vos? ¡A las computadoras les encanta hacer cuentas! Mirá —dijo Antonia, y sus dedos sacaron vibrantes sonidos al teclado mientras modificaba las líneas de Cecilia:

```
1 $fn = 200;
2 sphere(r=16);
3 translate([0,0,16+8])
4   sphere(r=8);
5 translate([0,0,16+2*8+4])
6   sphere(r=4);
7 translate([0,0,16+2*8+2*4+2])
8   sphere(r=2);
9 translate([0,0,16+2*8+2*4+2*2+1])
10  sphere(r=1);
11 translate([0,0,16+2*8+2*4+2*2+2*.5])
12  sphere(r=.5);
13 translate([0,0,16+2*8+2*4+2*2+2*1+2*.5+.25])
14  sphere(r=.25);
```

A Cecilia le pareció muy conveniente enterarse de que OpenSCAD aceptaba las operaciones matemáticas básicas como modo de expresión, pero sinceramente no veía cómo eso mejoraba su texto.

—Lo que quería era mostrarte cómo OpenSCAD acepta las operaciones matemáticas básicas como modo de expresión —confirmó Antonia, como si leyera su mente palabra por palabra—. En este caso particular, admito que no mejora sensiblemente tu texto; sin embargo, explicitar las operaciones matemáticas puede sugerirnos ideas, realzando el patrón que se encuentra detrás de ellas.

Antonia hizo un silencio que pudo parecer teatral, pero no lo fue: volvió a sumergirse en el texto de Cecilia para indagar en él una nueva reescritura.

7.2. VARIABLES

—Claramente, los radios de las esferas están firmemente vinculados: a partir de la primera, las demás tienen un tamaño igual a la mitad de la anterior. Si vos decidieras construir otra torre de esferas a partir de una inicial con radio diferente, deberías modificar las doce líneas de texto, cuando en realidad debería bastar (desde el punto de vista de la lógica de la construcción de la torre) con cambiar un solo dato: el radio de la primera esfera —razonó Antonia, y a Cecilia no le costó nada estar de acuerdo.

»Para eso existen las *variables*: ‘etiquetas’ que representan un valor, por ejemplo numérico. Mirá —dijo Antonia, mientras volvía a hacer tamborilear el teclado:

```
1 $fn = 200;
2 radio = 16;
3
4 sphere(r=radio);
5 translate([0,0,radio+radio/2])
6   sphere(r=radio/2);
7 translate([0,0,radio+2*radio/2+radio/4])
8   sphere(r=radio/4);
9 translate([0,0,radio+2*radio/2+2*radio/4+radio/8])
10  sphere(r=radio/8);
11 [...]
```

»Una vez que definís la variable `radio`, cada aparición de la misma en el texto es reemplazada por su valor; en nuestro caso, 16.

Cecilia, una vez más y atraída ahora por un nuevo concepto que sentía suavemente que le abría posibilidades que apenas podía sospechar por el momento, entrecerró los ojos y se inclinó hacia el monitor, mientras leía y asentía ligeramente con la cabeza.

—Está bueno —susurró.

7.3. EN BUSCA DE UNA LEY

—¿No es cierto? —asintió Antonia con una enorme sonrisa de satisfacción—. Ahora, si cambiás de idea acerca del tamaño de la base de la torre sólo debés modificar la línea número 2.

Tras un breve instante, en el que volvió a considerar el texto recién modificado, continuó:

—Sin embargo, aún se puede mejorar. Como ya dijimos, el radio y la altura de cada esfera están determinados por el radio de la primera; lo único que resta decidir por nuestra parte es cuántas esferas queremos. A partir de ahí, es pura mecánica: escribir las respectivas líneas de texto cuidando de respetar la lógica matemática que hay detrás de esa configuración en torre. Pero, ¿no son las computadoras ideales para el trabajo mecánico y repetitivo? ¿Y qué pasa si querés enhebrar, digamos, 25 esferas? ¿Será posible que debamos escribir 50 líneas de código virtualmente repetido? Es más, ¿no es sumamente probable que una escritora se equivoque al escribir tediosas líneas de contorno prácticamente idéntico? Las computadoras, por otra parte, no se distraen ni cometen erratas.

Antonia se detuvo, pensando evidentemente que su discurso no podía tener más que una sola conclusión. Cecilia sintió que debía decir algo, pero sólo acertó a levantar las cejas, invitándola a seguir.

—Pues bien —prosiguió Antonia, un tanto contrariada por no haber logrado el efecto buscado—; para lograr que la computadora realice por su cuenta la columna de esferas deseada debemos buscar antes la ley matemática que la sintetiza. Y para eso, una vez más, a mí me sirve mucho reescribir las respectivas expresiones.

```
1 $fn = 200;  
2 radio = 16;  
3  
4 translate([0,0,radio*0])
```

```
5     sphere(r=radio/1);
6 translate([0,0,radio*3/2])
7     sphere(r=radio/2);
8 translate([0,0,radio*9/4])
9     sphere(r=radio/4);
10    translate([0,0,radio*21/8])
11    sphere(r=radio/8);
12    translate([0,0,radio*45/16])
13    sphere(r=radio/16);
14 [...]
```

Cecilia se irguió de golpe en su silla:

—¡Hey! ¿Qué hiciste? —preguntó.

—Saqué factor común para cada altura —contestó Antonia, sorprendida—: $\text{radio}+2*\text{radio}/2+\text{radio}/4$ es igual a $\text{radio}*9/4$, ¿no?

—Sí, sí, claro —Cecilia se ruborizó ligeramente—. Pero también agregaste al principio una traslación nula: `translate ([0, 0, radio*0])`. No sólo eso: al radio de la primera esfera lo dividiste por 1: `sphere(r=radio/1)`... ¿Qué sentido tiene todo eso?

—Tiene muchísimo sentido —se defendió Antonia—. Estamos buscando un patrón matemático que abarque a *todas* las esferas, y eso incluye a la primera.

Cecilia asintió, comprendiendo, y trató de seguir el hilo de pensamiento propuesto por Antonia:

—La progresión de los radios es obvia: caen según potencias de 2: 1, 2, 4, 8, 16, etc. —afirmó.

—¡Tal cual! —asintió Antonia—. Y las potencias de 2 pueden escribirse como 2^i , donde i es la posición en la progresión, siempre y cuando empecemos a contar desde 0. Así, el primer radio es igual a 2^0 , ya que $2^0 = 1$.

—¡Qué conveniente que cualquier número elevado a la 0 sea igual a 1..! —acotó Cecilia con una sonrisa, y las dos rieron ligeramente recordando sus tiempos de estudiantes, cuando pocos profesores les explicaban la razón profunda de esa “rareza”.

—Muy bien —prosiguió Antonia—; ya resolvimos los radios. Ataquemos ahora las alturas. La sucesión tiene esta pinta: radio $\times 0$, radio $\times \frac{3}{2}$, radio $\times \frac{9}{4}$, radio $\times \frac{21}{8}$, radio $\times \frac{45}{16}$...

—Los denominadores son fáciles... —se adelantó Cecilia.

—Tal cual —confirmó Antonia—; más de lo mismo. Pero los numeradores no son tan obvios.

—¡Múltiplos de 3! —aventuró Cecilia—: 3×0 , 3×1 , 3×3 , 3×7 , 3×15 ...

—¡Exacto! —exclamó Antonia—. ¿Y qué onda esos factores que multiplican a 3: 0, 1, 3, 7, 15...?

Cecilia y Antonia estuvieron mirando la serie concentradas, hasta que la primera disparó:

—¡Son una unidad menos que los téminos de la sucesión de las potencias de 2!: $0 = 2^0 - 1$, $1 = 2^1 - 1$, $3 = 2^2 - 1$, $7 = 2^3 - 1$, $15 = 2^4 - 1$, ... ¡Las potencias de 2 nos persiguen!

Las dos rieron de buena gana, no sabemos si por la gracia del chiste ñño o por la alegría de sentir que resolvieron el secreto que una progresión de esferas enhebradas guardaba tan celosamente.

—Pasando en limpio, entonces —sentenció Antonia—; las alturas de las esferas siguen la ley $\frac{\text{radio} \times 3 \times (2^i - 1)}{2^i}$, donde i representa la ubicación de cada una en la sucesión, contando desde 0, y radio es el radio de la primera esfera.

—Y los radios respetan esta otra ley: $\frac{\text{radio}}{2^i}$ —agregó Cecilia, satisfecha.^{1,2,3}

—Y por hoy ya adelantamos mucho, Cecilia —soltó Antonia

¹En rigor, lo que Cecilia y Antonia hallaron es una regularidad manifestada en unos pocos téminos; no están justificadas para extender su aplicación a cualquier valor de i . Sin embargo, considero que dicha regularidad es bastante convincente en sí misma, por lo que estimo que podemos confiar en ella.

²La confianza no reemplaza a un teorema; esas fórmulas deben ser demostradas matemáticamente antes de ser aplicadas, para no hablar de publicarse en un libro, aun cuando se trate de uno tan ínfimo como éste. (Nota del Editor)

³⊗ (Nota de Cecilia, Antonia y Luis)

con un suspiro—. Creo que es mejor que sigamos en el capítulo siguiente, donde aprovecharemos estas flamantes leyes para que la computadora escriba por nosotras el texto completo de la torre.

Cecilia se preguntó a qué se refería Antonia con “el capítulo siguiente”, pero estaba demasiado entusiasmada y cansada como para preguntar.

Análisis literario – II

8.1. POTENCIAS

—MUY BIEN—arrancó Antonia—; nosotras queremos, entonces, generar ocho esferas cuyos radios sean iguales a $\frac{\text{radio}}{2^i}$ y sus alturas con respecto al origen sean iguales a $\frac{\text{radio} \times 3 \times (2^i - 1)}{2^i}$, donde i representa la ubicación de cada esfera en la sucesión, contando desde 0. En otras palabras, nos gustaría poder repetir algo como esto para cada valor de i entre 0 y 7:

```
1 translate([0,0,radio*3*(pow(2,i)-1)/pow(2,i)])
2 sphere(r=radio/pow(2,i));
```

Cecilia se detuvo a considerar el texto. En primer lugar, Antonia reemplazó la altura (el tercer componente de la indicación `translate`) por lo que parecía una expresión matemática: `radio * 3 * (pow(2,i)-1) / pow(2,i)`. Lo único desconocido por ella hasta el momento era la palabra `pow`; supuso que tenía que ver con la potenciación.

Antonia pareció leer su mente una vez más:

—Para expresar potencias en OpenSCAD se emplea la función `pow`¹; la sintaxis genérica es `pow(base,potencia)`. Por esa razón,

¹Quizá se deba a que en inglés potencia se dice *power*. (Nota del Editor)

`pow(2,i)` es matemáticamente igual a 2^i .

»Ahora bien —recapituló Antonia—; deberíamos entonces repetir las dos líneas del texto anterior asignando a `i` los valores que van desde 0 hasta 7. En otras palabras, necesitamos crear una variable `i` que tome *todos* esos valores en sucesión, en lugar de uno solo.

8.2. BUCLES

Antonia hizo un silencio pretendidamente teatral antes de continuar:

—Para lograr tan mágico y conveniente resultado nos valemos de la instrucción `for`:

```
1 radio = 16;
2
3 for (i = [0,1,2,3,4,5,6,7])
4   translate([0,0,radio*3*(pow(2,i)-1)/pow(2,i)])
5   sphere(r=radio/pow(2,i));
```

Cecilia casi no podía creerlo. «¿Tan fácil era?», se preguntó. Según le pareció entender, la indicación `for` asignaba a `i`, uno tras otro, los números encerrados entre corchetes, para con ellos crear las esferas correspondientes.

—Se puede mejorar un poco más —Antonia interrumpió sus pensamientos—. ¿No te parece molesto tener que escribir explícitamente cada uno de los números de la elemental sucesión aritmética que va de 0 a 7? No sólo es un poco molesto, sino que si quisieras hacerlo con 50 esferas, deberías escribir... demasiado, ¿no?

Cecilia estaba tan contenta con lo conseguido hasta aquí que bien pudiera haber escrito la sucesión de 0 a 100 entera; pero entendió el punto de Antonia.

—Pues bien, eso se puede lograr; mirá —prosiguió Antonia, cuya sonrisa crecía mientras escribía con Cecilia:

```

1 radio = 16;
2
3 for (i = [0:7])
4   translate([0,0,radio*3*(pow(2,i)-1)/pow(2,i)])
5     sphere(r=radio/pow(2,i));

```

»Cinco líneas de texto, contando el espacio para separar visualmente las ideas —la voz de Antonia no podía reflejar mejor la satisfacción que la embargaba, y Cecilia sintió que se contagiaba de la misma—; y si querés cambiar el tamaño de la base, o la cantidad de esferas, sólo debés modificar un valor: el 16 o el 7.

Antonia seguía leyendo el texto en silencio, como si buscara alguna posible mejora, frunciendo los labios e inclinando la cabeza a uno y otro lado. Finalmente dijo:

—Mirá, sólo para que lo tengas presente como posibilidad, te cuento: quizás esas expresiones matemáticas un tanto extensas puedan resultarte, cuando vuelvas al texto tras unos días, un tanto oscuras. Podemos considerar la posibilidad de escribirlas usando variables... a ver:

```

1 radio = 16;
2
3 for (i = [0:7]) {
4   ri = radio/pow(2,i);
5   zi = radio*3*(pow(2,i)-1)/pow(2,i);
6   translate([0,0,zi])
7     sphere(r=ri);
8 }

```

»Recordá que la asignación de cada variable debe concluir con un punto y coma. Además, y debido a que ahora el bucle `for` afecta a más de una instrucción (en este caso son tres: dos para las variables auxiliares y una para la creación del objeto trasladado) deben usarse llaves, a fin de encerrar con ellas el contenido que

debe repetirse. Cuando el `for` afecta a una sola instrucción pueden usarse también, pero no es necesario —sentenció Antonia.

Cecilia se moría de ganas de ver una torre de cincuenta esferas:

```

1 $fn = 200;
2 radio = 100;
3
4 for (i = [0:49]) {
5     ri = radio/pow(2,i);
6     zi = radio*3*(pow(2,i)-1)/pow(2,i);
7     translate([0,0,zi])
8     sphere(r=ri);
9 }
```

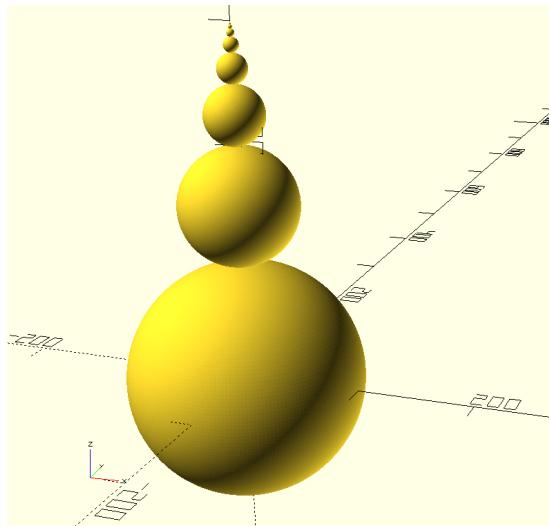


Figura 8.1: Cincuenta esferas apiladas por Cecilia usando un bucle. Lamentablemente, las progresiones geométricas se precipitan a valores ínfimos demasiado rápidamente, por lo que las últimas esferas no pueden apreciarse... ¡Aunque están ahí!

—La última y terminamos por hoy —prometió Antonia—;

cuando declarás un bucle usando un rango, no es necesario pasar escrupulosamente por cada uno de sus valores; es posible indicar un ‘salto’ entre uno y otro. De esta forma, si vos escribís `for(i=[0:2:7])`, i toma los valores 0, 2, 4 y 6; en otras palabras, vas avanzando de a 2:

```

1 $fn = 200;
2 radio = 16;
3
4 for (i = [0:2:7]) {
5   ri = radio/pow(2,i);
6   zi = radio*3*(pow(2,i)-1)/pow(2,i);
7   translate([0,0,zi])
8     sphere(r=ri);
9 }
```

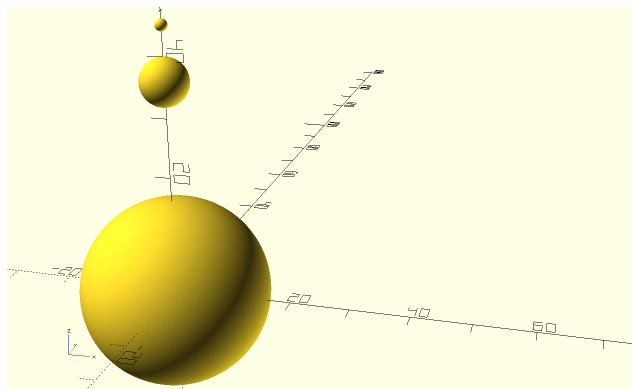


Figura 8.2: Tres esferas alternadas.

»La sintaxis genérica es `for(i=[inicio:paso:final])`. Por supuesto, el valor final puede quedar excluido del rango si el paso es demasiado grande; tal el caso que nos ocurrió a nosotras, donde el valor ‘7’ resultó omitido, ya que al ‘6’ debería haber seguido el ‘8’.

8.3. DESAFÍO MATEMÁTICO

Cecilia estaba feliz con la última versión del texto. Empezaba a comprender porqué a Antonia le gustaba tanto programar. Ahora era ella la que, casi sin proponérselo, leía y releía el texto en busca de posibles mejoras. Súbitamente su atención recayó en los valores numéricos empleados en las definiciones de r_i y z_i : esos '2' y '3' estaban claramente influidos por su voluntad inicial de construir una torre donde cada esfera tuviera un radio exactamente igual a la *mitad* de la inmediata inferior... Eso tuvo como efecto, además, que más allá de la octava o novena esfera las demás resultaran imperceptibles. «¿No sería genial que pudiera cambiar esos valores para obtener una torre que respetara otra progresión geométrica cualquiera?» —pensó—. «¿Y que esos valores dependieran de uno solo, que pudiera declararse al principio del texto como una variable y en función del cual se calcularan los r_i y z_i correspondientes..? De esa forma, si el factor de decrecimiento fuera, por ejemplo, 1,5, las esferas se parecerían más entre sí y la torre se angostaría con mayor suavidad».

Cecilia confió a Antonia sus ideas, y ambas trataron de encontrar unas fórmulas que las concretaran. Aún no lo consiguieron.²

8.4. UNA SOLUCIÓN DEMASIADO SOFISTICADA

Al día siguiente, Cecilia encontró a Antonia con gesto adusto y distante. Preocupada, intentó indagar si algo malo le ocurría. Tras mucho insistir, Antonia finalmente le confió su inquietud:

—Cecilia, logré que la torre crezca con un factor de decrecimiento cualquiera, a elección del usuario —el tono grave y serio

²El autor de este librito confiesa encontrarse en la misma melancólica situación.

de Antonia no condecía con lo que a Cecilia le parecía una noticia excelente.

—¡Genial! —Cecilia trató de sonar festiva, pero el gesto de su amiga la cohibía un poco. Tras unos instantes, agregó suavemente—: Antonia, ¿no deberías estar un poco más contenta?

—Pues... sí, es verdad —Antonia, de mala gana, consintió—. El problema es que usé una técnica un poco... sofisticada. No me siento preparada para explicártela aún. Debemos compartir antes otros recursos lingüísticos más elementales y básicos —concluyó, con tono de culpa.

Cecilia nunca dejaba de sorprenderse con Antonia: ¿eso la había preocupado?

—Pero, che..! ¡Parece mentira! —riendo, intentó tranquilizarla—. Dale, al menos contame cómo lo pensaste matemáticamente.

—Ah, bueno, eso sí —Antonia recobró, de pronto, su entusiasmo—. Como no fui capaz de hallar una fórmula de tipo ‘cerrada’, a la manera de la que conseguimos para el caso particular de que las esferas decrecieran con un factor igual a 2, traté de encontrar otra que reprodujera la expresión completa de la suma con la que empezamos; ¿te acordás cómo venía? —Antonia la escribió en el pizarrón de su oficina:

$$\text{Una esfera: } zi = 0$$

$$\text{Dos esferas: } zi = r + \frac{r}{k}$$

$$\text{Tres esferas: } zi = r + \frac{r}{k} + \frac{r}{k} + \frac{r}{k^2}$$

$$\text{Cuatro esferas: } zi = r + \frac{r}{k} + \frac{r}{k} + \frac{r}{k^2} + \frac{r}{k^2} + \frac{r}{k^3}$$

»En estas fórmulas ‘r’ es el radio de la esfera inicial y ‘k’ el factor de decrecimiento, que en tu texto original era igual a 2. Si te fijás, con cada nueva esfera se agregan dos términos; dejame escribir una suerte de fórmula ‘general’, en la que uso paréntesis

para destacar los términos que se agregan con cada esfera:

$$z_i = r \cdot \left[0 + \left(\frac{1}{k^0} + \frac{1}{k^1} \right) + \left(\frac{1}{k^1} + \frac{1}{k^2} \right) + \left(\frac{1}{k^2} + \frac{1}{k^3} \right) + \dots \right] \quad (8.1)$$

»Pues bien —y aquí el gesto de Antonia volvió a ensombrecerse—, lo que hice a continuación fue escribir una función que precisamente va sumando esos términos en OpenSCAD para cada valor de i .

—¡Guau! ¿Eso se puede hacer? ¡Excelente! —Cecilia ensayó un esfuerzo para conseguir que no decayera el entusiasmo de Antonia—. ¿Lo puedo ver?

Antonia se mostró inmediatamente muy esquiva; pareció querer encerrarse en sí misma. Cecilia sabía que si eso pasaba, sería muy difícil recuperarla:

—Dale —rogó—; mostrame esa solución...

—Tengo miedo —fue la lacónica respuesta que escapó de los labios de Antonia, casi a regañadientes.

—Pero... ¿De qué? —Cecilia preguntó, también con temor.

Antonia tardó en responder y, cuando lo hizo, con un hilo de voz, sus ojos parecían querer escapar por la ventana de su oficina hacia los jardines de Harvard:

—Pues... de que sientas que no entendés nada, te frustres, y salgas corriendo de este libro.

Cecilia la miró unos instantes que parecieron una eternidad. Luego, una sonrisa iluminó lentamente su rostro mientras decía, con un marcado ceceo:

—Dale, zonza... moztrame eze texto de una vez...

Antonia, desconcertada, pareció despertar bruscamente de un sueño, hasta que la sonora risa de Cecilia la contagió.

—Bueno, está bien —dijo Antonia, cuando sus risas finalmente se apagaron—; pero conste que por ahora será sólo un texto que podrás copiar y pegar sin entender, nada más. Te prometo, eso sí, que más adelante y cuando hayamos visto otros temas

más urgentes, vamos a volver sobre él y desvelar juntas todo su significado. Sólo te adelanto que en el texto, la variable `factor` se corresponde con la 'k' de las fórmulas anteriores —Antonia se detuvo dubitativa unos momentos, y agregó con un tono ligero— : ¡Quién sabe! quizás incluso te resulte divertido jugar con los valores de las variables involucradas, a ver qué onda.

```

1 $fn = 200;
2 radio = 16;
3 factor = 1.2;
4 for (i = [0:1:47]) {
5   ri = radio/pow(factor,i);
6   zi = z(i);
7   translate([0,0,zi])
8     sphere(ri);
9 }
10 function z(i,acc=0) =
11   (i==0)
12   ? radio*acc
13   : z(i-1,acc=acc+(factor+1)/pow(factor,i));

```

A pesar de las advertencias de Antonia, Cecilia hizo un esfuerzo por comprender sola el texto anterior. No le sorprendió comprobar que no lo conseguía. Lo mejor que logró fue identificar oscuramente la expresión `(factor+1)/pow(factor,i)` con esta otra: $\frac{k+1}{k^i}$, y ésta a su vez con el i-ésimo paréntesis de la fórmula (8.1): $(\frac{1}{k^{i-1}} + \frac{1}{k^i})$. Sobre lo demás se cernía un velo opaco, demasiado pesado como para descorrerlo. Pero eso no la desanimó: antes bien, el suave contorno del párrafo escrito entre las líneas 10 y 13, aunque secreto y oscuro, lanzaba destellos como un desafío a sus ojos. Su corazón de científica volvió a encenderse ante lo desconocido, y la pulsión por entender la hizo vibrar una vez más.

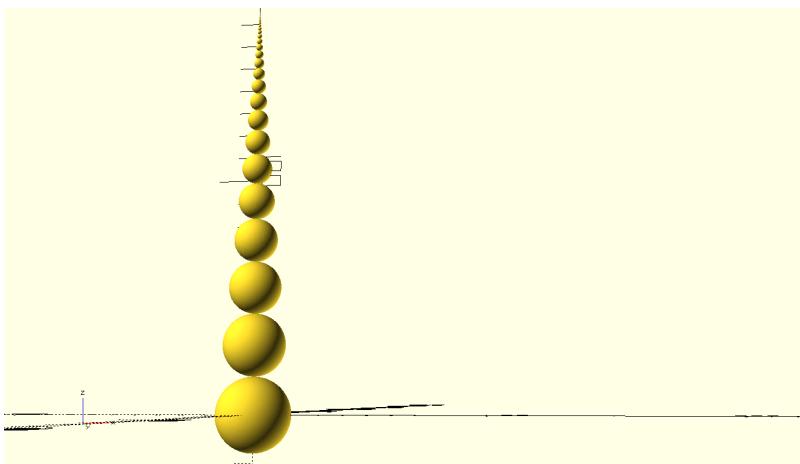


Figura 8.3: Una torre de muchas esferas.

— 9 —

Rotación de objetos

—¿T_E GUSTA? —Antonia recibió a Cecilia al día siguiente con la sonrisa de quien hace un regalo.

```
1 $fn = 200;
2 // piernas
3 translate([10,0,0])
4   cylinder(h=40,r1=3,r2=6);
5 translate([-10,0,0])
6   cylinder(h=40,r1=3,r2=6);
7 // torso
8 translate([-18,-7,40])
9   cube([36,14,40]);
10 // brazos
11 translate([23,0,45])
12   cylinder(h=30,r1=3,r2=5);
13 translate([-23,0,45])
14   cylinder(h=30,r1=3,r2=5);
15 // hombros
16 translate([23,0,75])
17   sphere(r=5);
18 translate([-23,0,75])
19   sphere(r=5);
20 // cabeza
21 translate([0,0,89])
22   sphere(r=10);
```

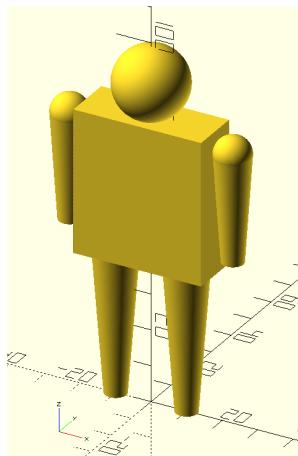


Figura 9.1: Una colección de objetos que a Cecilia se le antojó un robotito.

9.1. COMENTARIOS Y VARIANTES DEL CILINDRO

A Cecilia le pareció simpático: —¿Es un robotito? —preguntó.

—Ponele —concedió Antonia—. Pero además me permite mostrarte dos características más del lenguaje, que pueden ser de tu interés. Por un lado, cualquier línea que comience con dos barras invertidas juntas (//) es ignorada por OpenSCAD; por esa razón, podés aprovecharla para anotar en ella comentarios: pequeñas aclaraciones para un eventual lector humano. Y eso —Antonia miró fijamente a Cecilia— te incluye a vos: te prometo que tras unos días de no convivir con tu texto vas a agradecer esas pequeñas notas al margen.

»Por otro lado —continuó con pesado tono didáctico—, los cilindros admiten, al momento de ser creados, la declaración de dos radios: `r1` (que representa el de la base) y `r2` (el de la ‘tapa’).

—Interesante —acotó Cecilia, tomando a su cargo el teclado—.

¿Entonces puedo crear un cono anulando el radio superior de un cilindro?

```
1 $fn = 200;
2
3 cylinder(h=50, r1=20, r2=0);
```

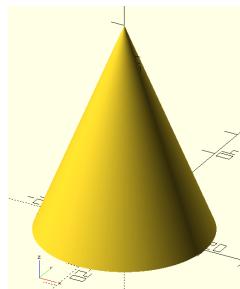


Figura 9.2: Cecilia descubre cómo escribir un cono.

—Parece que sí —aprobó Antonia con una sonrisa de satisfacción al comprobar que Cecilia estaba apropiándose de una de las ideas fundamentales y más fértiles de la programación: hacerle continuamente preguntas a la computadora bajo la forma de texto, y aprendiendo de sus respuestas. No resultaba raro en una científica, después de todo: ellas se dedicaban a interrogar la naturaleza con experimentos, leyendo y desvelando sus secretos en los resultados.

9.2. ROTACIÓN

»¿No te parecería bueno que nuestro simpático robot fuera capaz de mover sus brazos? Para eso debemos ver cómo es posible rotar objetos en OpenSCAD—Antonia nunca fue muy hábil para introducir temas nuevos de manera natural—. Creo que no te resultará sorprendente que eso se logre con la indicación `rotate`.

»`rotate([x,y,z])` indica cómo debe ser rotado el objeto al cual se aplica: ‘x’ son los grados rotados alrededor del eje homónimo, y lo mismo vale para los otros dos ejes. En el caso del ejemplo que escribí en la figura 9.3, el cilindro (originalmente creado vertical)

9. ROTACIÓN DE OBJETOS

```
1 $fn = 200;  
2  
3 rotate([90,0,0])  
4 cylinder(h=30,r=5);
```

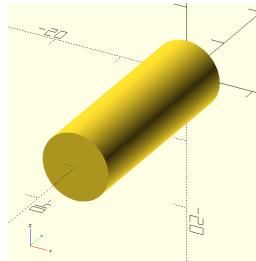


Figura 9.3: Antonia rota un cilindro 90° alrededor del eje X.

es rotado 90° en torno del eje X. El ejemplo de la figura 9.4 hace lo propio alrededor del eje Y, pero usando sólo 45° .

```
1 $fn = 200;  
2  
3 rotate([0,45,0])  
4 cylinder(h=30,r=5);
```

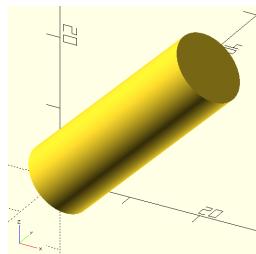


Figura 9.4: Antonia rota otro cilindro, esta vez 45° en torno al eje Y.

»Por supuesto, podés indicar más de un valor distinto de 0 en la rotación; pero en ese caso tené en cuenta que primero se aplica la rotación en X, luego la rotación en Y, y por último la rotación en Z. En la figura 9.5, por ejemplo, roté el cilindro 45° alrededor del eje X, luego 70° en torno al Y, y finalmente 10° respecto del Z. Hago hincapié en el orden —Antonia sintió la necesidad de aclarar— ya que, como bien sabés, las rotaciones no son comutativas.

```

1 $fn = 200;
2
3 rotate([45,70,10])
4 cylinder(h=30,r=5);

```

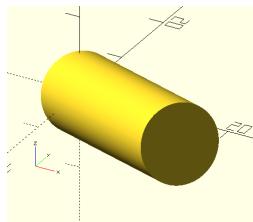


Figura 9.5: Antonia aplica una rotación alrededor de los tres ejes coordinados a otro cilindro más.

9.3. APLICACIÓN DE VARIAS TRANSFORMACIONES

Cecilia prefirió no comentar que, efectivamente, ya lo sabía.
—¿Y cómo hago para elegir yo misma el orden de las rotaciones?
—cuestionó, dando por descontado que era posible.

—En ese caso, y en otros similares, podés ir acumulando una transformación detrás de otra; mejor dicho, ‘antes’ que otra —Antonia rió levemente de lo que sin duda consideró un chascarrillo oportuno—. La idea es que un objeto transformado es considerado *otro* objeto en sí mismo, que puede a su vez ser transformado, y así *ad infinitum*; mirá:

```

1 $fn = 200;
2
3 rotate([0,10,0])
4 rotate([0,0,-30])
5 rotate([120,0,0])
6 cylinder(h=30,r=5);

```

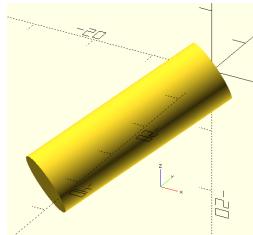


Figura 9.6: Antonia aplica tres rotaciones en sucesión a... otro cilindro.

»En este párrafo —abundó Antonia— primero roto el cilindro original 120° alrededor del eje X. El resultado es un nuevo objeto

9. ROTACIÓN DE OBJETOS

que roto -30° en torno al Z. Finalmente, ese nuevo objeto es rotado 10° en el eje Y.

9.4. ROTACIONES Y TRASLACIONES

Cecilia no estaba segura de entender. Sintió que, para hacerlo plenamente, debía escribir algo que afirmara sus incipientes sospechas e intuiciones. Le gustó la idea de que un objeto transformado fuera otro objeto, susceptible de ser nuevamente transformado. Recordó que una traslación es una transformación. Una idea destelló entonces en su mente; tomó el teclado mientras, sin darse cuenta, mascullaba:

—¿Qué pasaría si...?

```
1 $fn = 200;  
2  
3 // esfera de r=1  
4 rotate([0,0,-90])  
5 translate([20,0,0])  
6 sphere(r=1);  
7  
8 // esfera de r=2  
9 translate([20,0,0])  
10 rotate([0,0,-90])  
11 sphere(r=2);
```

Cecilia contempló el resultado de su texto con atención en la figura 9.7. Había escrito dos esferas de distinto tamaño para poder distinguirlas y así facilitar la comparación. A ambas las afectó con las mismas transformaciones (una rotación alrededor del eje Z y una traslación a lo largo del eje X); lo que cambió es el orden: la esfera de radio igual a 1 primero fue trasladada y luego rotada, al revés de lo que hizo con la segunda esfera.

Antonia estuvo a punto de interrumpir sus pensamientos con una de sus explicaciones, pero Cecilia la anticipó:

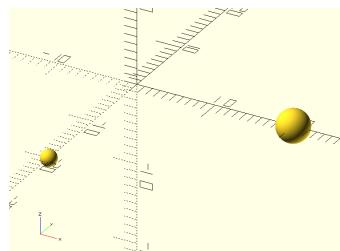


Figura 9.7: Cecilia explora la posibilidad de trasladar y rotar dos esferas.

—¡Pará! —dijo con tono que no admitía réplica—. Creo que ya lo tengo —y volvió a sumergirse en su propio texto.

En primer lugar, comentó casi todas las líneas del mismo, dejando sólo la primera esfera con la traslación. «Buen truco» —pensó, satisfecha de su astucia—, «así puedo concentrarme en un paso a la vez».

```

1 $fn=200;
2
3 // esfera de r=1
4 //rotate([0,0,-90])
5   translate([20,0,0])
6     sphere(r=1);
7
8 // esfera de r=2
9 //translate([20,0,0])
10 // rotate([0,0,-90])
11 //   sphere(r=2);
```

«Muy bien» —se felicitó, mientras seguía con sus cavilaciones—; «ahora vamos a rotar ese nuevo objeto».

```

1 $fn=200;
2
3 // esfera de r=1
4 rotate([0,0,-90])
```

9. ROTACIÓN DE OBJETOS

```
5   translate([20,0,0])
6     sphere(r=1);
7
8 // esfera de r=2
9 //translate([20,0,0])
10 //  rotate([0,0,-90])
11 //    sphere(r=2);
```

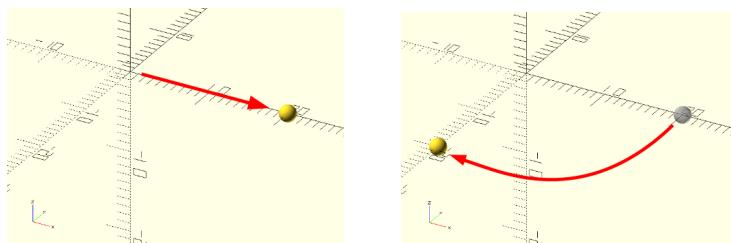


Figura 9.8: Esfera trasladada y luego rotada.

Cecilia pudo analizar el resultado de su texto en la imagen derecha de la figura 9.8. «Así que las rotaciones siempre refieren al origen de coordenadas: si el objeto está lejos de él, es rotado manteniendo su distancia respecto al mismo» —Cecilia sentía que estaba empezando a entender—. «Por otro lado, la esfera 2 primero fue rotada»:

```
1 $fn=200;
2
3 // esfera de r=1
4 //rotate([0,0,-90])
5 // translate([20,0,0])
6 //   sphere(r=1);
7
8 // esfera de r=2
9 //translate([20,0,0])
10   rotate([0,0,-90])
11     sphere(r=2);
```

«Lo cual, en el caso de una esfera, es ocioso» —Cecilia rió para sus adentros, divertida—. «¡Una esfera, después de todo, tiene simetría esférica! Y si ahora la trasladamos, llegará sencillamente a la misma posición que hubiera alcanzado sin esa espuria rotación previa»:

```

1 $fn=200;
2
3 // esfera de r=1
4 //rotate([0,0,-90])
5 // translate([20,0,0])
6 // sphere(r=1);
7
8 // esfera de r=2
9 translate([20,0,0])
10 rotate([0,0,-90])
11 sphere(r=2);

```

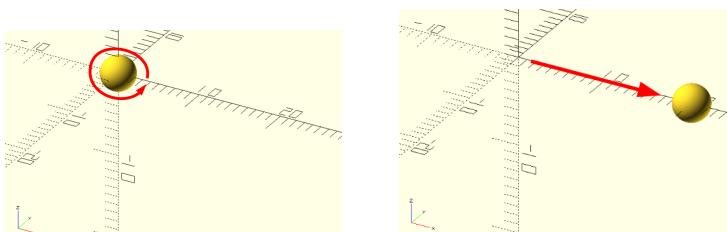


Figura 9.9: Esfera rotada (inútilmente) y luego trasladada.

9.5. ROTACIONES, TRASLACIONES Y BUCLES

Antonia aplaudió lentamente, con gesto de aprobación:

—¿Puedo decir algo ahora, o la Sra. Grace Hopper está demasiado enfrascada en sus grandes piezas literarias? —dijo, con gesto algo burlón.

9. ROTACIÓN DE OBJETOS

Cecilia salió de su ensimismamiento algo ruborizada; sintió la necesidad de replicar:

—Cómo vos gustéis, Augusta Ada, condesa de Lovelace —dijo, inclinando suave y ceremoniosamente la cabeza.

Ambas rieron ligeramente; les gustaba intercambiarse los nombres de científicas célebres. Antonia tomó el teclado y empezó a escribir. Mientras lo hacía, miraba a Cecilia de soslayo.

```
1 $fn=200;  
2  
3 rotate([0,0,0])  
4 translate([20,0,0])  
5 sphere(r=2);  
6 rotate([0,0,30])  
7 translate([20,0,0])  
8 sphere(r=2);  
9 rotate([0,0,60])  
10 translate([20,0,0])  
11 sphere(r=2);  
12 rotate([0,0,90])  
13 translate([20,0,0])  
14 sphere(r=2);  
15 rotate([0,0,120])  
16 translate([20,0,0])  
17 sphere(r=2);
```

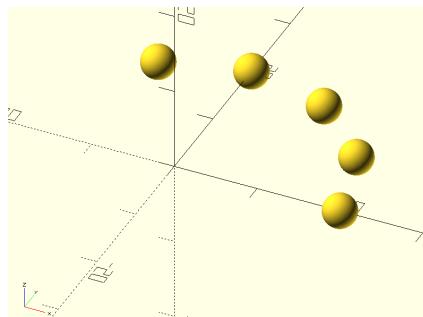


Figura 9.10: Antonia escribe varias esferas esperando que una idea surja naturalmente en Cecilia.

Cecilia seguía con la vista las líneas de Antonia. De pronto, soltó en un rapto de entusiasmo:

—¡Pará! ¡Ya sé lo que estás haciendo! A ver, dejame a mí... —y sin esperar que le respondiera, tomó el teclado para sí. Había notado un patrón en el nuevo texto: Antonia estaba creando esferas idénticas, separadas del centro de coordenadas la misma

distancia y luego rotadas alrededor del eje Z con ángulos múltiplos de 15° . Esa repetición regular gritaba la necesidad de un bucle:

```

1 $fn=200;
2
3 for (alfa=[0:15:359])
4   rotate([0,0,alfa])
5   translate([20,0,0])
6   sphere(r=2);

```

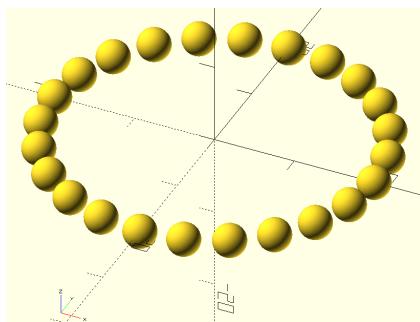


Figura 9.11: Cecilia descubre cómo escribir un anillo de esferas.

Cecilia giró su rostro en dirección a Antonia; ambas se miraron radiantes de alegría. Cecilia volvió a mirar el monitor; comprobó que su intuición funcionaba: había creado un bucle que asignaba a la variable `alfa` valores entre 0 y 359 en intervalos de 15, y con ellos creaba esferas trasladadas la misma distancia y rotadas dicho ángulo `alfa`. Por momentos parecía demasiado hermoso para ser verdad.

—Hermoso —Antonia confirmó sus pensamientos—. Ahora, ¿te animás a hacer algo como la figura 9.12?

Cecilia pegó un respingo en su silla. Trató de describir en palabras lo que veía. «Parecen ser varios anillos de esferas como el mío, pero de radios diferentes» —pensó. Tras unos momentos

9. ROTACIÓN DE OBJETOS

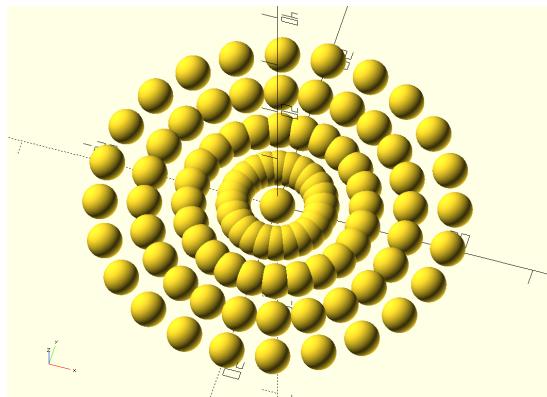


Figura 9.12: Anillos de esferas

de concentrada contemplación, sintió que la solución pasaba por utilizar otro bucle.

—¡Vaamooss! —exclamó Cecilia con aire triunfal, y alzando los brazos mientras contemplaba su solución en la figura 9.13. Comprobó con alegría que su sospecha funcionó: las líneas 4 a 7 de su texto creaban, como antes, un nuevo objeto (un anillo de esferas), y la línea 3 declaraba una nueva variable (x) que

```
1 $fn=200;  
2  
3 for(x=[20:-5:0])  
4   for (alfa=[0:15:359])  
5     rotate([0,0,alfa])  
6       translate([x,0,0])  
7         sphere(r=2);
```

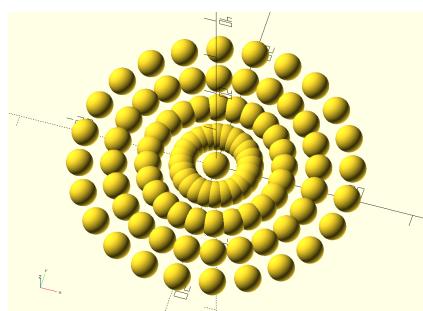


Figura 9.13: Solución de Cecilia para producir anillos de esferas.

tomaba valores entre 20 y 0, a intervalos de 5 (el intervalo debía ser negativo porque x debía ir disminuyendo), con los cuales creaba anillos de distinto radio.

Cecilia sintió que su mente comenzaba a poblararse de posibilidades innumerables y variadas, y que sus dedos no alcanzarían nunca a recorrerlas todas. «Esto está buenísimo» —pensó.

—Te pido dos más, y terminaremos por este capítulo —prometió Antonia, cuya sonrisa delataba que compartía la misma alegría que Cecilia—. Hacé un cilindro de anillos de esferas, y un cono de la misma naturaleza. Te adelanto que este último es bastante más difícil...

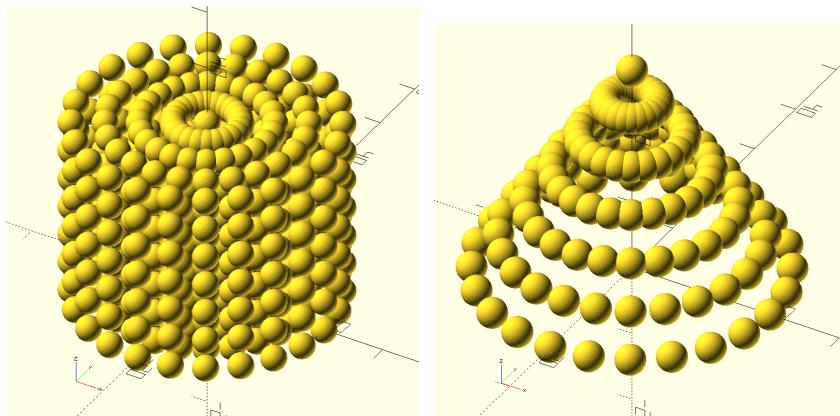


Figura 9.14: Antonia propone a Cecilia escribir un cilindro y un cono de esferas.

A Cecilia le costó bastante el cono, aunque descubrió —para su sorpresa— que sólo requería el empleo de dos bucles, mientras que para el cilindro necesitó tres.

9.6. CILINDRO DE ESFERAS: UNA SOLUCIÓN

El cilindro no le resultó demasiado difícil a Cecilia; comprendió que no se trataba de otra cosa que una repetición a lo largo del eje Z del anillo que ya había conquistado. Eso le sugirió el empleo de un bucle más.

```
1 $fn=200;  
2  
3 for(z=[0:5:30])  
4   for(x=[20:-5:0])  
5     for(alfa=[0:15:359])  
6       rotate([0,0,alfa])  
7       translate([x,0,z])  
8       sphere(r=2);
```

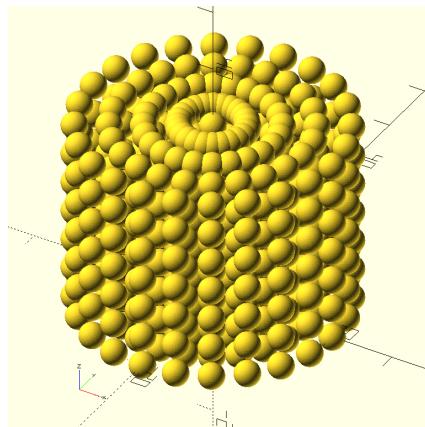


Figura 9.15: Solución de Cecilia al cilindro de esferas.

Las líneas 4 a 8 de la figura 9.15 eran una mera copia de las que utilizó para escribir el anillo en la sección anterior, con una salvedad: en la línea 7, el '0' que indicaba la traslación en el eje Z fue reemplazado por la variable 'z', cuyo valor es determinado por el bucle de la línea 3: en él, se la hacía recorrer los valores de 0 a 30, en intervalos de a 5, creando de esa manera siete pisos de anillos.

9.7. CONO DE ESFERAS: UNA SOLUCIÓN

El cono representó un verdadero desafío: Cecilia comprendió que no sólo debía crear anillos de altura creciente, sino de radio

decreciente. Sin embargo, tras pensarlo un rato se convenció de que la altura de cada anillo (z , digamos) y su correspondiente radio (x , ¿por qué no?) no eran independientes: a la altura máxima (z_{\max} , ¿soy original?) le correspondía el radio mínimo (de hecho, un radio nulo), mientras que a la altura mínima (también de valor nulo, al yacer en el plano XY) le correspondía el radio máximo (x_{\max} , ya que estamos). Evocando sus recuerdos de Álgebra¹ llegó a una relación satisfactoria entre los valores de x y z :

$$x = x_{\max} - z \cdot \left(\frac{x_{\max}}{z_{\max}} \right)$$

```

1 $fn=200;
2 for(z=[0:5:30]) {
3   x=20-z*20/30;
4   for (alfa=[0:15:359])
5     rotate([0,0,alfa])
6     translate([x,0,z])
7     sphere(r=2);
8 }
```

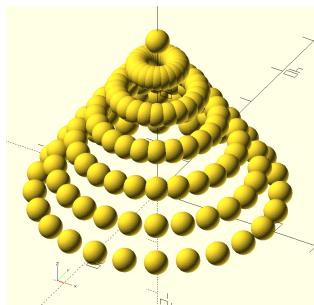


Figura 9.16: Solución de Cecilia al cono de esferas.

En primer lugar, entonces, declaró en la línea 3 el bucle que hace que z recorra las alturas de cada piso. Con ese valor, calculaba entonces en la línea 4 el correspondiente radio x del mismo. Y con ambos valores, creaba el debido anillo en las líneas 5 a 8. Las llaves de las líneas 3 y 9 resultaban necesarios, puesto que el bucle de la línea 3 abarcaba más de una instrucción.

¹El lector interesado y armado con la noción matemática de función puede deducirla sin más que considerar una relación lineal vinculando ambos valores.

— 10 —

Vectores, rangos y más bucles

10.1. MÁS BUCLES

ANTONIA APROBÓ con una amplia sonrisa el trabajo de Cecilia: —Sólo para que lo tengas en cuenta —aclaró—, OpenSCAD te permite declarar más de una variable en el mismo bucle; por ejemplo, tu solución al cilindro puede escribirse también así:

```
1 $fn=200;  
2  
3 for(z=[0:5:30],  
4     x=[20:-5:0],  
5     alfa=[0:15:359])  
6     rotate([0,0,alfa])  
7     translate([x,0,z])  
8     sphere(r=2);
```

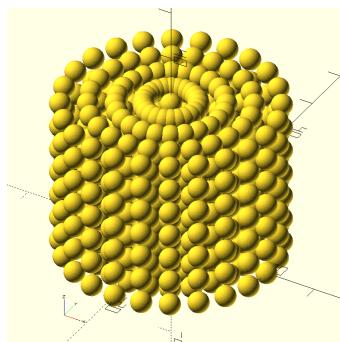


Figura 10.1: Antonia declara un `for` con más de una variable.

Cecilia no estaba segura de que esta reescritura mejorara su texto, pero decidió que una nueva posibilidad expresiva siempre merecía ser tenida en cuenta.

10.2. VECTORES

—Un tipo de valor muy útil en OpenSCAD es el denominado *vector* —dijo Antonia—, que no es otra cosa que una colección ordenada de otros valores. Se escribe encerrando estos entre corchetes y separados por comas; por ejemplo, el vector [23,12,71] tiene tres elementos: el 23, el 12 y el 71.

Cecilia reconoció de inmediato esa sintaxis:

—¡Estuvimos usando vectores todo el tiempo! En `translate`, `rotate` y para indicar el tamaño de un cubo, por ejemplo.

—Tal cual —acordó Antonia—. Y ahora, sabiendo que los vectores no son otra cosa que valores particulares en sí mismos, podemos guardarlos en variables también:

```
1 v = [10 ,20 ,30] ;
2
3 rotate(v)
4     translate(v)
5         cube(v);
```

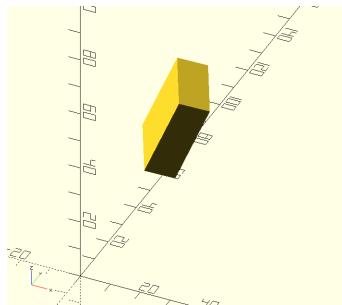


Figura 10.2: Antonia abusa de las posibilidades de los vectores.

A Cecilia el ejemplo le pareció muy tonto, pero captó la idea. Antonia prosiguió:

—Es importante que tengas en cuenta que entre los valores que un vector admite se encuentran, también, los propios vectores; es decir, podemos tener vectores de vectores. En ese caso solemos hablar de *matrices*. Nos resultarán utilísimas cuando escribamos nuestro reloj de Sol digital. Un ejemplo posible es `[[1,2,3],[4,5,6],[7,8,9]]`, vector cuyos elementos son los vectores `[1,2,3]`, `[4,5,6]` y `[7,8,9]`.

```

1 v = [[10,20,30],
2     [40,-10,-15],
3     [0,-90,0]];
4 rotate(v[2])
5 translate(v[1])
6 cube(v[0]);
7 // Lo anterior es
8 // equivalente a:
9 //rotate([0,-90,0])
10 // translate([40,-10,-15])
11 // cube([10,20,30]);

```

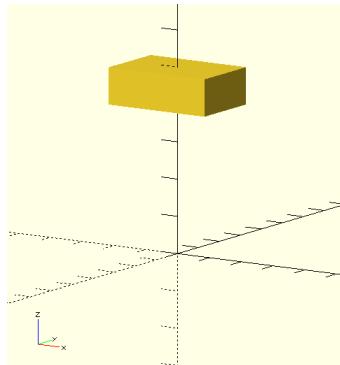


Figura 10.3: Antonia usa un vector de vectores (también conocido como matriz) y se refiere a sus elementos mediante índices.

»En un vector es importante el orden de los elementos — continuó Antonia—: cada uno tiene su posición particular. Decimos que el primero está en la posición ‘0’, el segundo en la posición ‘1’, y así siguiendo. La razón de comenzar en ‘0’ tiene que ver con la manera en que almacenan los números las computadoras; comenzar en ‘1’ supondría desperdiciar memoria. En fin, es muy técnico —concluyó Antonia agitando suavemente en el aire la mano derecha, como si el conjuro “es muy técnico” bastara para dar algo por explicado.

»Es posible referirse a un elemento en particular de un vector adosándole, al final, el apéndice $[i]$, donde i representa la posición del elemento de nuestro interés. Así, $[4,5,6][0]$ es el elemento 4, $[[1,2,3],[4,5,6],[7,8,9]][1]$ vale tanto como $[4,5,6]$, etc. A ver si con el ejemplo de la figura 10.3 queda más claro. Como comentario en las líneas 7 a 11 te puse un código equivalente al escrito en las líneas 1 a 6.

Cecilia seguía sintiendo que se trataba de un ejemplo muy tonto, pero se conformó con entender el punto.

10.3. RANGOS

—Los rangos que venimos utilizando en los bucles podrían ser contemplados como vectores ‘comprimidos’; quiero decir, a una le gustaría pensar que algo como `[0:2:10]` equivale al vector `[0,2,4,6,8,10]` —Antonia seguía disertando, con su molesto tono didáctico—. Pero la semejanza es superficial; por ejemplo, escribir algo como `[0:2:10] [3]`, en lugar de devolvernos el valor 6, nos tira un error. Lo que sí admiten los rangos es ser aludidos mediante una variable; así, es posible escribir un cubo de esferas de esta limpia manera —agregó, señalando la figura 10.4.

```
1 $fn= 50;  
2  
3 s = [0:10:100];  
4  
5 for(x=s,y=s,z=s)  
6     translate([x,y,z])  
7         sphere(2);
```

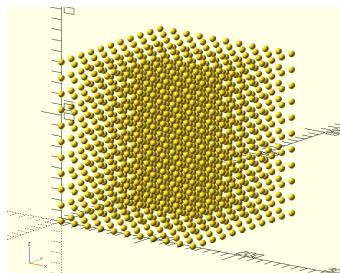


Figura 10.4: Antonia asigna un rango a una variable a modo de ejemplo.

Este texto a Cecilia le gustó un poco más; comprendió que, en el fondo, era equivalente al de la figura 10.5. La diferencia entre ambos era que si una quería modificar el tamaño del cubo, en el primer caso bastaba con cambiar el valor de la variable `s` en la línea 3, mientras que en el segundo era preciso modificar las líneas 3 a 5.

De todos modos, no pudo evitar considerar que, en ejemplos tan minúsculos como estos, las diferencias terminaban diluyéndose en una cuestión de gustos. Pero en todo caso le parecía que siempre resultaba útil contar con más de una forma de escribir lo mismo: una nunca sabía con qué problemas se enfrentaría más tarde.

```
1 $fn= 50;  
2  
3 for(x=[0:10:100] ,  
4     y=[0:10:100] ,  
5     z=[0:10:100])  
6     translate([x,y,z])  
7         sphere(2);
```

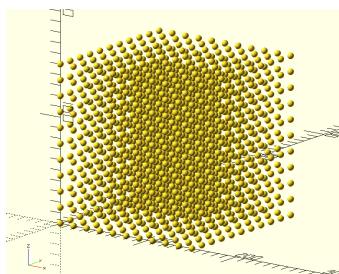


Figura 10.5: Cecilia desarrolla el ejemplo anterior de Antonia, para asegurarse de haber entendido el concepto.

de, y disponer de un mayor número de herramientas lingüísticas siempre era mejor.

Una torre más o menos medieval – I

CUANDO CECILIA entró al día siguiente en la oficina de Antonia, la encontró frente a un nuevo objeto en el monitor de la computadora.

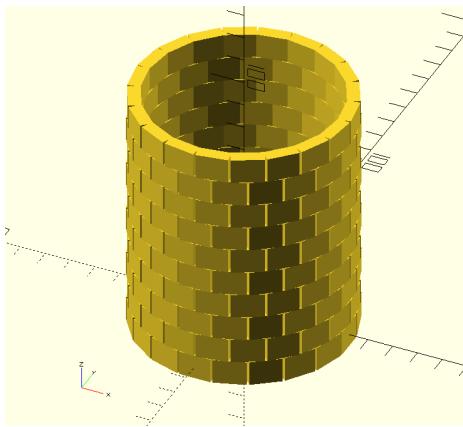


Figura 11.1: Una torre más o menos medieval.

—¿Con qué estás jugando ahora, Antonia? —preguntó con una sonrisa, mientras se sentaba a su lado.

—Quería introducirte un nuevo concepto, que considero fundamental a esta altura del manual —empezó Antonia, para asom-

bro de Cecilia que nunca supo que estaba participando de un “manual”—. Hasta ahora estuvimos escribiendo a partir de los objetos primitivos que OpenSCAD nos ofrece; sin embargo, el lenguaje también nos brinda la posibilidad de crear nuestros propios objetos ‘básicos’ (y no tan básicos), a fin de usarlos como cualquier otro en nuestra descripción de una escena.

Cecilia sintió de golpe que siempre supo, en el fondo, que esa posibilidad *debía* existir.

—Para mostrarte cómo se concreta esa posibilidad, se me ocurrió que podemos crear algo como esto —continuó Antonia señalando el monitor con el mentón—: Una torre que evoca recuerdos más o menos medievales. Mi idea es que la construyamos en tres pasos, definiendo para ello tres nuevos ‘objetos’, que dependerán cada uno del anterior: el objeto ‘ladrillo’, el objeto ‘piso’ y el objeto ‘torre’. Es decir, vamos a escribir la torre como un objeto hecho de ‘pisos’, y cada piso como un objeto hecho de ‘ladrillos’.¹

11.1. MÓDULOS

»La manera de crear la definición de un objeto es con la palabra **module**.

EL LADRILLO

Cecilia sintió la incómoda sensación de que Antonia se estaba burlando de ella: ¡La pantalla aparecía vacía! Si bien no dijo nada, su expresión debió haber sido lo suficientemente elocuente como para que hasta Antonia se diera cuenta:

¹El improbable lector de este librito no debe dejarse confundir con el uso impreciso e informal que hace Antonia del témino “objeto” en un contexto computacional: los objetos a los que hace referencia nada tienen que ver con los que pueblan lenguajes como Smalltalk, C++ o Java, entre otros. (Nota del Editor)

```

1 module ladrillo(size) {
2     cube(size, center=true);
3 }

```

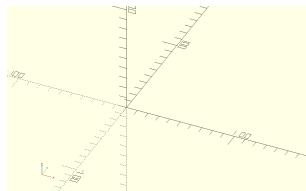


Figura 11.2: Antonia define el módulo ladrillo.

—Es natural que nada haya aparecido —la tranquilizó con una sonrisa—; tan sólo hemos definido el objeto; aún no lo hemos usado.

»La palabra `module` debe ir seguida por el nombre que queremos darle al nuevo objeto. Inmediatamente después deben ir, entre paréntesis, los parámetros que determinarán el mismo: en nuestro caso, su tamaño². Podría ocurrir, aunque es raro, que necesitemos un objeto cuya determinación sea fija e inamovible: en ese caso aún debemos escribir los paréntesis, sólo que estarán vacíos. Finalmente, escribiremos entre llaves las acciones que harán posible nuestro objeto: en nuestro caso, y por simplicidad, decidí que fuera un mero cubo, cuyo tamaño sea el parámetro de la definición, y esté centrado en el origen.

Cecilia seguía en silencio y mirando el monitor; no estaba segura de comprender.

Antonia prosiguió:

—Ahora sí podemos usar nuestro flamante ladrillo —y escribió las líneas de la figura 11.3.

Cecilia creía empezar a entender, pero algo no le cerraba:

—Antonia, ¿para qué definís un objeto que ya existe? Al final, la definición de ladrillo sólo hace referencia a un cubo común y silvestre.

²Nos hubiera encantado usar la palabra “tamaño” en el código, pero por alguna razón OpenSCAD se niega a aceptar variables con una “ñ”... ¡Y nos negamos rotundamente a estampar la palabreja “tamano”, ni siquiera en este manualcito!

```
1 ladrillo([20,10,5]);  
2 translate([25,0,0])  
3 ladrillo([10,8,20]);
```

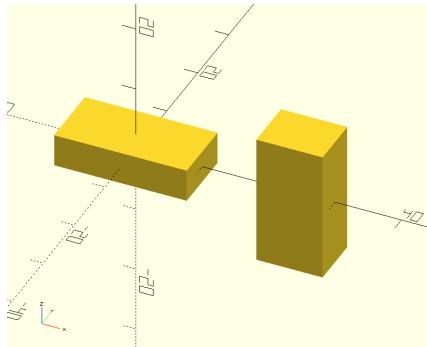


Figura 11.3: Antonia usa el módulo ladrillo

Antonia se revolvió en la silla; parecía impacientarse. Cecilia no supo en principio si era con ella, o consigo misma. El hecho de que Antonia apretara los labios, como quien busca la palabra o expresión justa, le hizo suponer que la impaciencia se debía a su constante angustia de sentir que no sabía explicarse debidamente. Ésa era una de las mayores preocupaciones de Antonia; preocupación a veces desmedida. Tal vez por eso la docencia la apasionara tanto, y tal vez por eso la padeciera tanto.

—En principio —comenzó lentamente Antonia—, fijate que al menos nos salvamos de indicar ‘center=true’ en cada uso del ladrillo. Pero eso no es lo más importante —Antonia volvió a detenerse, y suspiró—. El caso es que nosotras ahora podemos dedicarnos a definir y usar los otros objetos que necesitamos (el piso y la torre), siempre empleando para ellos el objeto ‘ladrillo’. Para esto, es verdad, podríamos usar sin problemas el objeto primitivo ‘cubo’. Pero ahora —Antonia se inclinó ligeramente hacia Cecilia, intensificando el tono de su voz—, imaginate que recién al final, con la torre efectivamente realizada, nos damos cuenta de que, no sé, sería mejor que los ladrillos tuvieran los bordes redondeados. O que su superficie fuera rugosa. O que presentaran

una textura ajedrezada. ¡Qué sé yo! —Antonia subrayó la idea alzando sus ojos al techo—. En ese caso (que descubrirás que surge muy naturalmente al escribir; hay cosas cuya oportunidad resulta aparente recién cuando el trabajo está muy avanzado) sólo deberemos cambiar la definición del módulo ‘ladrillo’, y no todas las apariciones de ‘cube’ que hubiéramos escrito a lo largo del texto —concluyó Antonia, escrutando en los ojos de Cecilia si la explicación había sido eficaz.

Cecilia pudo sentir un atisbo de comprensión, pero creía que sólo entendería cabalmente el concepto que tanto trabajo le costaba a Antonia trasmitir cuando lo viera realizado en un caso concreto. De todas formas, comprendió que eso sólo sería posible avanzando, por lo que decidió confiar en ella y aceptar, al menos provisoriamente, que era buena idea encerrar el concepto de ‘ladrillo’ en un módulo propio.

EL PISO

—Muy bien —dijo Antonia, tomando el silencio de Cecilia como un permiso para seguir—; ahora ataquemos el problema del piso. Mientras pensamos cómo hacerlo, podemos comenzar escribiendo el imprescindible esqueleto de su módulo.

```
1 module piso( ) {  
2  
3 }
```

—Pensemos ahora en los parámetros que lo definen: ¿Qué características determinan un piso de nuestra torre? —y Antonia se detuvo, cediendo la palabra tácitamente a Cecilia, quien se puso a pensar en voz alta:

—Hmmmm.... ¿su radio?

—De acuerdo —aprobó Antonia—; ¿y qué más?

A Cecilia le empezó a gustar el juego:

—¿El espesor de la pared? ¿Su alto?

—A favor —volvió a acordar Antonia—; ¿y qué más?

Cecilia cerró los ojos tratando de visualizar uno de los pisos de la torre. Tras breves instantes, exclamó:

—¡La cantidad de ladrillos que lo forman!

—Exacto —concluyó Antonia, y se puso a escribir:

```
1 module piso(  
2   radio,  
3   altura,  
4   espesor,  
5   n_ladrillos) {  
6 }
```

—A mí, personalmente —confesó Antonia—, y a medida que escribo estructuras más complejas, me gusta detallar los parámetros con nombres más elocuentes que ‘r’, ‘e’ o ‘n’. En ese caso, y para que no me queden líneas abigarradas y difíciles de leer, escribo un parámetro por línea. Pero es cuestión de gustos; también podrías escribir el módulo así:

```
1 //      PISO  
2 // r: radio  
3 // a: altura  
4 // e: espesor  
5 // n: cantidad de  
6 //      ladrillos  
7 module piso(r,a,e,n) {  
8 }
```

Cecilia no se sentía especialmente inclinada, al menos por el momento, hacia ninguna de las dos formas en particular.

—Ahora sí —Antonia prosiguió—, debemos escribir, dentro de las llaves, la forma en que efectivamente se construye un piso —Antonia hizo una pausa, y señalando el teclado a Cecilia, preguntó:

—¿Te animás?

Cecilia no sabía muy bien por dónde empezar, pero tomó el teclado aparentando, lo mejor que pudo, seguridad.

«Muy bien» —pensó—, «tengo que disponer una serie de ladrillos en círculo. ¡Pero ya hice algo parecido! Fue con esferas. No debería ser tan distinto: crear un ladrillo, alejarlo del centro, rotarlo un cierto ángulo, y repetir lo mismo con otros ladrillos, usando para eso un bucle. A ver...»

```
1 module piso(  
2     radio,  
3     altura,  
4     espesor,  
5     n_ladrillos) {  
6     for (alfa=[0:?;359])  
7         rotate([0,0,alfa])  
8         translate([radio,0,0])  
9         ladrillo([espesor,?,altura]);  
10    }  
11 }
```

Cecilia, al terminar de escribir lo anterior, sintió lo que siente todo aquél que explora un problema escribiéndolo: el sonido sordo que produce sondear los límites de la propia comprensión del mismo. Sintió, por un lado, que estaba bien encaminada: cada ladrillo debía alejarse del centro una distancia igual al radio del piso (línea 8), y luego rotarse un cierto ángulo alfa (línea 7). Además, dos de las dimensiones de cada ladrillo venían determinadas por los parámetros de dicho piso: su espesor y altura. Pero había dos datos cuya ignorancia sólo descubrió al intentar describir textualmente el piso: el largo de los ladrillos, y el ángulo que debían ser rotados. Sospechó que ambos valores dependían del parámetro aún no empleado por ella: `n_ladrillos`.

Antonia parecía seguir el hilo de sus pensamientos:

—Si tuvieras que conquistar sólo una enseñanza de estos capítulos, tal vez debería ser ésta: ‘Nada aclara tanto las ideas como ponerse a escribirlas’. Y ése es quizás el mayor mérito y

virtud de la programación: obligarnos a escribir nuestras ideas con la máxima escrupulosidad, la mayor claridad posible, la más delicada prolijidad y una rigurosa intolerancia a cualquier forma de ambigüedad. ¡Si supieras la cantidad de problemas que entendí mientras los escribía! Mejor dicho: ¡gracias a que los escribía! —Antonia suspendió su pequeño discurso con un tono de voz apenas exaltado. Quizá por primera vez desde que la conocía, Cecilia no pensó que exageraba.

Volvió a sumirse en el problema: «Bien, tengo que calcular el incremento de alfa y el largo del ladrillo. El incremento es fácil: si hay que encasar $n_{\text{ladrillos}}$ en una vuelta completa de 360° , dicho incremento tiene que ser igual a $\frac{360^\circ}{n_{\text{ladrillos}}}$. Voy a escribirlo en la línea 6:»

```
1 module piso(
2     radio,
3     altura,
4     espesor,
5     n_ladrillos) {
6     i_alfa=360/n_ladrillos;
7     for(alfa=[0:i_alfa:359])
8         rotate([0,0,alfa])
9         translate([radio,0,0])
10        ladrillo([espesor,?,altura]);
11 }
```

«Ahora me falta el largo de cada ladrillo» —pensó—. «¡Ya se! Todos los ladrillos deben formar el perímetro del piso; el mismo es igual $2 \times \pi \times \text{radio}$. Y en ese perímetro deben caber $n_{\text{ladrillos}}$. Por lo tanto, cada uno debe medir $\frac{2 \times \pi \times \text{radio}}{n_{\text{ladrillos}}}$. Lo definiré en la línea 7:»

```
1 module piso(
2     radio,
3     altura,
4     espesor,
5     n_ladrillos) {
```

```
6     i_alfa=360/n_ladrillos;
7     largo=2*3.1416*radio/n_ladrillos;
8     for (alfa=[0:i_alfa:359])
9         rotate([0,0,alfa])
10        translate([radio,0,0])
11        ladrillo([espesor,largo,altura]);
12 }
```

—Hermoso, Cecilia... hermoso —Antonia estaba radiante—.

De paso te cuento que OpenSCAD cuenta entre sus valores básicos una aproximación muy buena de π : la escribís ‘PI’, así, en mayúsculas.

Cecilia estaba ansiosa por comprobar que su piso funcionaba (de paso tuvo en cuenta la sugerencia de Antonia acerca del valor de PI en la nueva línea 7):

```
1 module piso(
2   radio,
3   altura,
4   espesor,
5   n_ladrillos) {
6   i_alfa=360/n_ladrillos;
7   largo=2*PI*radio/n_ladrillos;
8   for (alfa=[0:i_alfa:359])
9     rotate([0,0,alfa])
10    translate([radio,0,0])
11    ladrillo([espesor,largo,altura]);
12 }
13
14 piso(40,5,5,20);
```

Los aplausos de Cecilia, al contemplar la figura 11.4, debieron escucharse hasta en la oficina de Fumington.

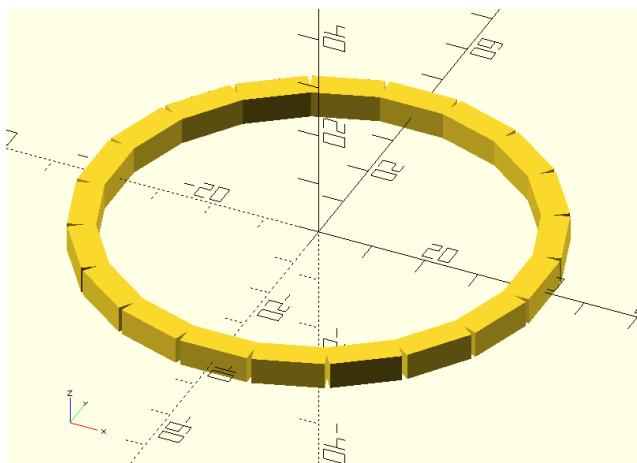


Figura 11.4: Un piso de ladrillos.

— 12 —

Una torre más o menos medieval – II

12.1. LOS PRIMEROS *bugs* DE CECILIA

—**H**AY UN PEQUEÑO detalle; mirá el piso bien desde arriba —advirtió Antonia.

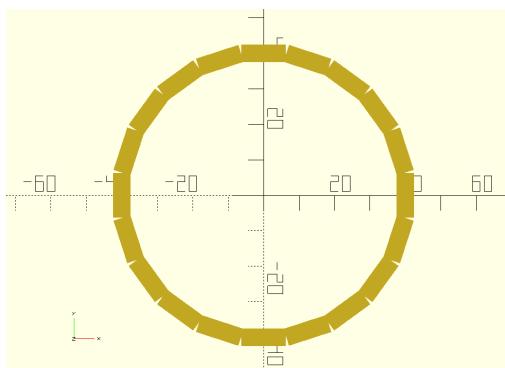


Figura 12.1: Antonia muestra el piso de ladrillos cenitalmente, a fin de que Cecilia detecte un *bug* sutil.

Cecilia seguía viéndolo hermoso.

—A ver, ¿qué me decís del radio? ¿Es de 40? —señaló Antonia con aire cómplice.

—¡Uy! ¡Tenés razón! —Cecilia concedió—. Me quedó un poco más grande... ¿Por qué será?

—Ah, no sé —Antonia puso cara de exagerada perplejidad—; el piso lo escribiste vos. Vos sabrás...

Cecilia aceptó el reto con una sonrisa. Para apreciar mejor el problema, hizo descender el piso por debajo de los ejes de coordenadas:

```
1 translate([0,0,-20])  
2 piso(40,5,5,20);
```

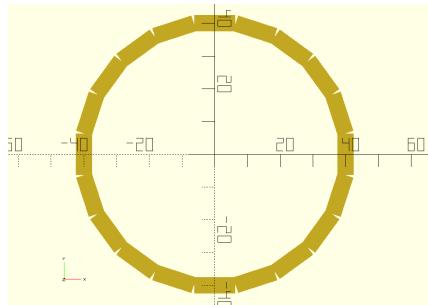


Figura 12.2: Cecilia desciende el piso a fin de encontrar su *bug* poniendo de manifiesto los ejes coordinados.

«Hmmm... La marca de los ‘40’ cae justo en el centro de un ladrillo» —pensó—. «Y, sin embargo, a cada uno lo desplacé exactamente 40mm con la instrucción `translate([radio, 0, 0])`... ¿Por qué no me queda el piso con un radio de 40mm?»

Cecilia estaba desorientada. Decidió volver a recorrer todo su texto, hasta que sus ojos se detuvieron en la definición del módulo `ladrillo`.

—¡Claro! —dijo, ahora en voz alta, para compartirlo con Antonia—. Si el ladrillo lo hago aparecer centrado, cuando lo traslado es su *centro* el que se aleja 40mm.

—Perfecto —aprobó Antonia—; ahora, ¡a corregirlo!

A Cecilia no le pareció muy difícil lograrlo, una vez que localizó el problema: se trataba de desplazar el ladrillo una distancia igual al radio del piso *menos* la mitad del espesor de su pared, lo

```

1  translate([0,0,-10])
2  ladrillo([5,15,5]);
3  translate([40,0,-10])
4  ladrillo([5,15,5]);

```

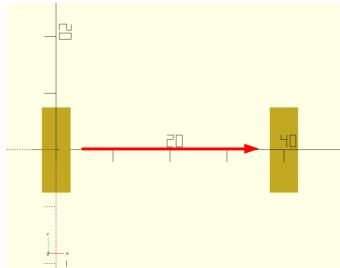


Figura 12.3: Cecilia descubre la razón de su *bug*.

cual se conseguía modificando la línea 10:

```

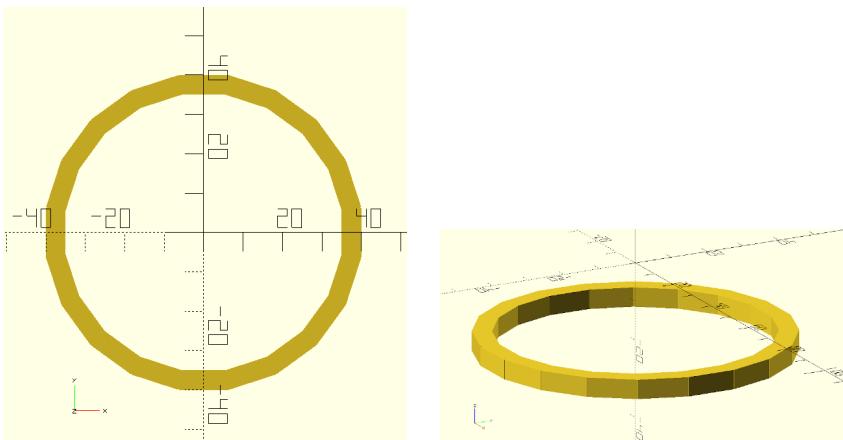
1  module piso(
2    radio,
3    altura,
4    espesor,
5    n_ladrillos) {
6    i_alfa=360/n_ladrillos;
7    largo=2*PI*radio/n_ladrillos;
8    for (alfa=[0:i_alfa:359])
9      rotate([0,0,alfa])
10     translate([radio-espesor/2,0,0])
11     ladrillo([espesor,largo,altura]);
12 }
13 translate([0,0,-20])
14 piso(40,5,5,20);

```

—¡Oh..! —el entusiasmo de Cecilia se enfrió de golpe frente a la figura 12.4: si bien el piso adquirió el radio deseado, habían desaparecido esos bonitos intersticios que le darían una apariencia tan característica y satisfactoria a su torre una vez terminada.

—Los ladrillos están tocándose unos con otros —reconoció Cecilia apesadumbrada.

—Maldita exactitud de la geometría! —masculló Antonia, con aire burlón, y pretendiendo darle así una pista del problema que



- (a) El piso parcialmente resuelto...
 (b) ...ya que, contrariamente a lo deseable, los ladrillos no presentan intersticios entre sí.

Figura 12.4: Cecilia casi soluciona el *bug* de su piso.

ahora aparecía.

Cecilia, tras unos instantes, replicó:

—Sí, entiendo: el largo de los ladrillos fue calculado precisamente para que llene el perímetro del piso. Así que necesito que sean un poco más chicos. No sé; a ver si modifico el cálculo de la variable `largo` en la línea 7, multiplicándolo por un número menor que 1...

Cecilia probó varios valores, hasta que encontró uno que le gustó:

```

1 module piso(
2   radio,
3   altura,
4   espesor,
5   n_ladrillos) {
6   i_alfa=360/n_ladrillos;
```

```
7     largo=2*PI*radio/n_ladrillos*0.95;
8     for (alfa=[0:i_alfa:359])
9       rotate([0,0,alfa])
10      translate([radio-espesor/2,0,0])
11      ladrillo([espesor,largo,altura]);
12   }
13
14 piso(40,5,5,20);
```

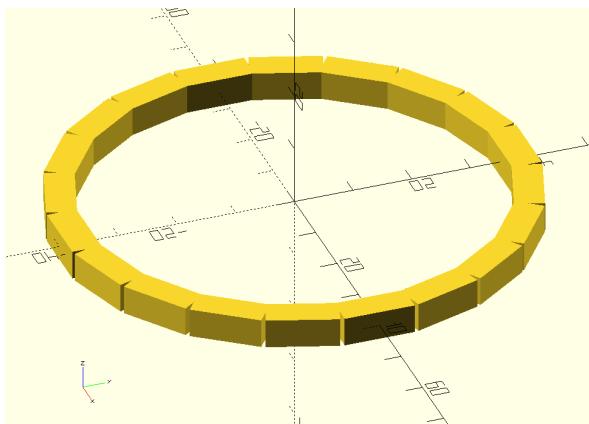


Figura 12.5: Los ladrillos, ahora, se presentan con intersticios visualmente gratos.

12.2. PARÁMETROS OPCIONALES

Cecilia, feliz por el resultado conquistado, giró para mirar a Antonia, a tiempo de advertir que un relámpago cruzaba por los ojos de su amiga. «¡Ups!» —pensó Cecilia— «¿Qué se le habrá ocurrido ahora?».

—Cecilia —empezó Antonia con un tono que presagiaba más trabajo—; ¿estás segura de que los lectores de tu texto (entre los

que te contarás vos misma más adelante) estarán de acuerdo con el valor 0,95? ¿No te parece que si tienen derecho a elegir el radio, espesor, altura, etc., de la torre de sus sueños, no deberían también poder decidir la magnitud de los intesticios entre los ladrillos? —preguntó, con aire de no admitir otra respuesta que un rotundo “Sí”.

—Tenés razón —Cecilia no pudo menos que admitirlo—; pero la hacemos fácil: agregamos un parámetro más al módulo: `factor_inter`, y listo:

```
1 module piso(
2   radio ,
3   altura ,
4   espesor ,
5   n_ladrillos ,
6   factor_inter) {
7   i_alfa=360/n_ladrillos ;
8   largo=2*PI*radio/n_ladrillos*factor_inter ;
9   for (alfa=[0:i_alfa:359])
10    rotate([0,0,alfa])
11    translate([radio-espesor/2,0,0])
12    ladrillo([espesor,largo,altura]);
13 }
```

—No está mal —aprobó Antonia displicentemente—; pero se puede hacer mejor. Si bien estamos de acuerdo en que es deseable que el lector pueda elegir, a la hora de crear un piso, la magnitud de los intersticios, también es cómodo ofrecerle un valor ‘convencional’ y que ‘funciona’, en caso de que no quiera ponerse a buscar uno por sí mismo. Para eso resultan insustituibles los parámetros opcionales: parámetros que, si el usuario no los indica, toman un valor predeterminado; mirá cómo modiflico la línea 6 —dijo Antonia, tomando a su cargo el teclado.

```
1 module piso(
2   radio ,
3   altura ,
```

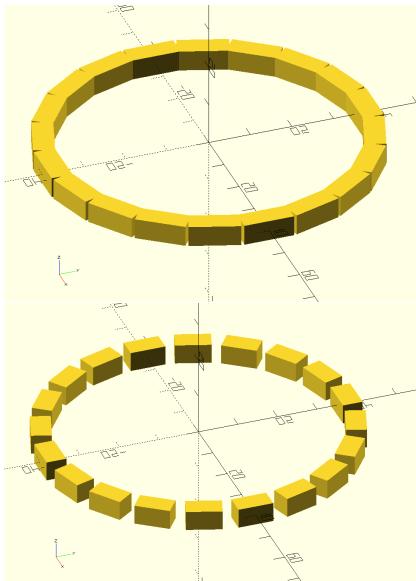
```

4   espesor ,
5   n_ladrillos ,
6   factor_inter=0.95) {
7     i_alfa=360/n_ladrillos;
8     largo=2*PI*radio/n_ladrillos*factor_inter;
9     for (alfa=[0:i_alfa:359])
10       rotate([0,0,alfa])
11       translate([radio-espesor/2,0,0])
12       ladrillo([espesor,largo,altura]);
13   }

```

»Ahora, si el usuario escribe `piso(40,5,5,20)` obtendrá un piso con ladrillos un 95 % (el valor ‘por defecto’) más pequeños; pero si escribe `piso(40,5,5,20,0.7)`, los ladrillos serán un 70 % más chicos:

`piso(40,5,5,20);`



`piso(40,5,5,20,0.7);`

Definitivamente, 0,7 era un valor muy bajo para los intersticios; pero Cecilia captó la idea y apreció las posibilidades que podían brindarle los parámetrosopcionales.

Una torre más o menos medieval – III

—Y A TENEMOS ladrillos y pisos —recapituló Antonia—; ahora definamos la torre.

Cecilia entendió inmediatamente que ese plural aludía a ella.

13.1. LA TORRE

Con la mirada vuelta hacia el monitor, y la imaginación puesta en el dibujo original de la torre que le había mostrado Antonia, Cecilia pensaba: «Bien, esto es bastante fácil: una torre no es más que una colección de pisos colocados uno encima del otro. Pero no te apures, Cecilia» —se dijo a sí misma—; «primero lo primero: escribamos el esqueleto del módulo torre»:

```
1 module torre ( ) {  
2  
3 }
```

«Excelente» —Cecilia trató de darse ánimos para continuar—. «¿Qué parámetros definen una torre? Claramente, su altura. ¡Ah! También la cantidad de pisos que tenga».

Si bien se obligó a seguir pensando, sólo pudo concluir que los demás parámetros que se le ocurrían pertenecían a cada piso: radio, espesor de las paredes, etc.

```
1 module torre (
2     altura,
3     n_pisos) {
4
5 }
```

«Muy bien; ahora supongo que debo crear un bucle que repita, para cada valor de n_pisos, un piso trasladado hacia arriba. Pero, ¿cuánto debo trasladarlos?» —Cecilia pensó un momento—. «¡Claro! Si tengo que repartir la altura entre n_pisos, cada piso medirá $\frac{\text{altura}}{\text{n_pisos}}$ de alto»:

```
1 module torre (
2     altura,
3     n_pisos) {
4     alto_piso=altura/n_pisos;
5     for (i=[0:n_pisos]){
6         z=i*alto_piso;
7         translate([0,0,z])
8             piso(40,
9                  alto_piso,
10                 5,
11                 20);
12     }
13 }
```

Cecilia no pudo evitar cierto orgullo mientras releía su texto. En la línea 4 calculaba la altura de cada uno de los pisos. En la definición del bucle de la línea 5, la variable i recorría cada uno de ellos. En la línea 6 calculaba la altura z que debían ser trasladados hacia arriba, y enseguida los creaba y trasladaba.

—Cecilia —Antonia interrumpió sus pensamientos con un tono que hizo que su compañera intuyera la inminencia de un problema—: ¿Por qué hiciste variar i desde 0 hasta n_pisos?

Cecilia respondió con sencillez:

—Porque quería que el primer piso estuviera... bueno... en el ‘piso’; para eso necesitaba que el primer valor de z fuera igual a

0, lo cual sólo podía conseguir haciendo que `i` comenzara en ese valor en lugar de 1 —Cecilia no podía ver ninguna fisura en su razonamiento.

Antonia replicó suavemente:

—En eso estoy completamente de acuerdo; lo que cuestiono es la cota superior...

Cecilia volvió a mirar, extrañada, el texto: ¿qué problema podía haber en usar `n_pisos` como máximo? ¿No era precisamente el tope? Tras unos instantes, se golpeó la frente suavemente con la palma de la mano:

—¡Por supuesto! ¡Si `i` va de 0 hasta `n_pisos`, la torre me va a quedar de `n_pisos+1` pisos! —Cecilia rio de buena gana de su desiste—. Nada más fácil de corregir: cambio la línea 5 y listo:

```

1 module torre (
2     altura,
3     n_pisos) {
4     alto_piso=altura/n_pisos;
5     for (i=[0:n_pisos-1]){
6         z=i*alto_piso;
7         translate([0,0,z])
8             piso(40,
9                 alto_piso,
10                5,
11                20);
12     }
13 }
```

Antonia miró el texto de Cecilia frunciendo los labios; Cecilia ya esperaba una inminente objeción revestida de sugerencia.

—Cecilia —comenzó Antonia—; ¿no te parece que quien recree una torre tiene derecho a elegir su radio, el espesor de sus paredes y la cantidad de ladrillos por piso? Quiero decir que esos valores, si bien son usados por el módulo `piso`, son características propias de la torre también, aun cuando no sean ‘procesadas’ por ella.

Cecilia pensó que lo dicho por Antonia tenía mucho sentido,

así que agregó esos parámetros a la torre a fin de que ella los pasara a cada piso:

```
1 module torre (
2     altura,
3     n_pisos,
4     radio,
5     espesor,
6     ladrillos_por_piso,
7     factor_inter=.95) {
8     alto_piso=altura/n_pisos;
9     for (i=[0:n_pisos-1]){
10         z=i*alto_piso;
11         translate([0,0,z])
12             piso(radio,
13                 alto_piso,
14                 espesor,
15                 ladrillos_por_piso,
16                 factor_inter);
17     }
18 }
```

13.2. PARÁMETROS POR NOMBRE

Cecilia tuvo que revisar la definición de piso para recordar en qué posición iba cada parámetro; si bien sabía que tenía un radio, un espesor de pared, etc., no estaba segura de cuál iba en primer lugar, en segundo lugar, etc.

Antonia, una vez más, pareció adivinar sus pensamientos:

—Hay una forma muy útil y práctica de pasar parámetros a un módulo sin depender de su posición: aludiéndolos por nombre. Mirá —y Antonia volvió a tomar el teclado:

```
1 module torre(
2     altura,
3     n_pisos,
```

```
4     radio ,
5     espesor ,
6     ladrillos_por_piso ,
7     factor_inter=.95) {
8     alto_piso=altura/n_pisos;
9     for (i=[0:n_pisos-1]){
10        z=i*alto_piso;
11        translate([0,0,z])
12          piso(radio=radio,
13                espesor=espesor ,
14                altura=alto_piso ,
15                n_ladrillos=ladrillos_por_piso ,
16                factor_inter=factor_inter);
17      }
18  }
```

—Intercambié a propósito la posición de las líneas 13 y 14 —Antonia guiñó un ojo a su amiga, suponiendo sin duda que acababa de parecer muy astuta—; el módulo piso funcionará igual, asignando a su propia variable radio el valor que con ese mismo nombre recibió la torre, dando a su propia variable altura el valor que la torre creó con el nombre alto_piso, etc. Es decir, usando los parámetros por nombre ya no importa en qué posición los uses; pero sí es importante, por supuesto, que uses los nombres correctos.

A Cecilia la última aclaración le pareció que estaba muy de más.

13.3. OTRO *bug* DE CECILIA

Mientras releían juntas el texto, Cecilia notó que Antonia sonreía con un gesto especial de reserva que conocía demasiado bien; supo inmediatamente que en su texto había otro error.

—Ok, Antonia —dijo mirándola fijamente a los ojos—; ¿me vas a decir o no en qué más me equivoqué?

Antonia se sonrojó ligeramente al verse sorprendida, pero reaccionó inmediatamente:

—Averigualo vos: creá una torre —lanzó, y en su sonrisa había ahora un desafío.

Cecilia no se lo hizo repetir; de paso, decidió crear la torre aludiendo explícitamente a sus parámetros:

```
1 torre(  
2   altura=100,  
3   radio=40,  
4   espesor=5,  
5   n_pisos=10,  
6   ladrillos_por_piso=20);
```

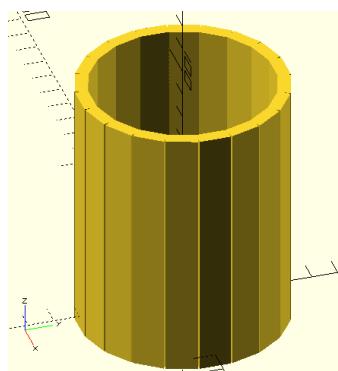


Figura 13.1: Los ladrillos de la flamante torre escrita por Cecilia no parecen apilarse como sería de desear.

—Nooo... —gimió Cecilia, cubriendo sus ojos con una mano—. ¡No me digas nada! —atajó inmediatamente a Antonia—. Ya sé: debo rotar los pisos para que cada ladrillo se apoye, por partes iguales, en dos ladrillos del piso inferior. Toda torre que se precie tiene que tener esa pinta.

Antonia asintió con una amplia sonrisa.

«Muy bien, Cecilia, ¡A pensar!» —se dijo a sí misma, mientras volvía a sumergirse en su texto—. «Cada piso, además de ser trasladado hacia arriba, debe ser rotado un cierto ángulo. Ahora bien, ¿cuánto vale ese ángulo?» —Cecilia se detuvo a pensar—. «¡Pero si ya hice una cuenta muy parecida! Es la mitad del ángulo que abarca un ladrillo en su piso; éste era igual a $\frac{360^\circ}{\text{ladrillos_por_piso}}$; así que el ángulo que debo rotar un piso para que se ‘corra’ como debe con respecto al anterior es $\frac{180^\circ}{\text{ladrillos_por_piso}}$ ».

```
1 module torre(
2     altura,
3     n_pisos,
4     radio,
5     espesor,
6     ladrillos_por_piso,
7     factor_inter=.95) {
8     alto_piso=altura/n_pisos;
9     delta_alfa=180/ladrillos_por_piso;
10    for (i=[0:n_pisos-1]){
11        z=i*alto_piso;
12        alfa=i*delta_alfa;
13        rotate([0,0,alfa])
14        translate([0,0,z])
15        piso(radio=radio,
16              espesor=espesor,
17              altura=alto_piso,
18              n_ladrillos=ladrillos_por_piso,
19              factor_inter=factor_inter);
20    }
21 }
```

Cecilia volvió a releer sus últimas modificaciones al texto: en la línea 9 definía la variable `delta_alfa` de acuerdo a la fórmula que dedujo momentos antes. Luego, en la línea 12, calculaba la variable `alfa`, con la que en la siguiente línea rotaba el piso correspondiente dicho ángulo. Le pareció lógico: cada piso debía

ser rotado `delta_alpha` grados con respecto al anterior, por lo que las rotaciones se iban ‘acumulando’: multiplicar `i` (el número de piso) por `delta_alpha` parecía la solución idónea.

Instintivamente miró a Antonia para descubrir en su gesto algún otro error que se le hubiera pasado por alto, pero la encontró con una sonrisa impenetrable. Decidió entonces jugársela:

```
1 torre(  
2   altura=100,  
3   radio=40,  
4   espesor=5,  
5   n_pisos=10,  
6   ladrillos_por_piso=20);
```

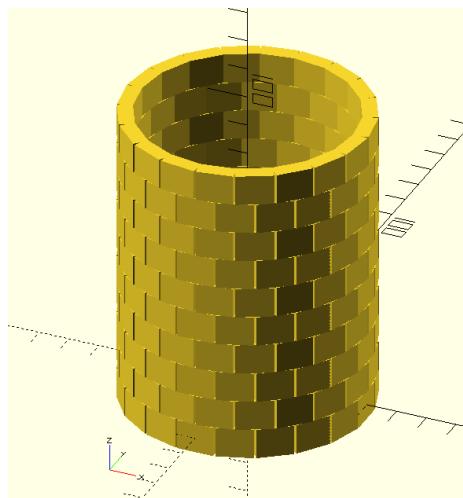


Figura 13.2: La torre finalmente conquistada.

¡Anduve! ¡Por fin!

Cecilia y Antonia se miraron con alarma: ¿quién había dicho eso?¹

¹Pido disculpas al lector: fui yo, naturalmente. El entusiasmo, usted entiende.

13.4. El texto completo, una apología y un poco de provocación

13.4. EL TEXTO COMPLETO, UNA APOLOGÍA Y UN POCO DE PROVOCACIÓN

Cecilia y Antonia volvieron su atención al texto completo creador de torres.

```
1 module ladrillo (size) {
2   cube(size,center=true) ;
3 }
4
5 module piso (
6   radio,
7   altura,
8   espesor,
9   n_ladrillos,
10  factor_inter=0.95) {
11    i_alfa=360/n_ladrillos;
12    largo=2*PI*radio/n_ladrillos*factor_inter;
13    for (alfa=[0:i_alfa:359])
14      rotate([0,0,alfa])
15      translate([radio-espesor/2,0,0])
16      ladrillo([espesor,largo,altura]);
17 }
18
19 module torre (
20   altura,
21   n pisos ,
22   radio,
23   espesor,
24   ladrillos_por_piso ,
25   factor_inter=.95) {
26   alto_piso=altura/n_pisos;
27   delta_alfa=180/ladrillos_por_piso;
28   for (i=[0:n_pisos-1]){
29     z=i*alto_piso;
```

Pero no le diga nada a ellas. Gracias.

```
30     alfa=i*delta_alfa;
31     rotate([0,0,alfa])
32     translate([0,0,z])
33     piso(radio=radio,
34           espesor=espesor,
35           altura=alto_piso,
36           n_ladrillos=ladrillos_por_piso,
37           factor_inter=factor_inter);
38   }
39 }
```

—¿Te das cuenta ahora de las posibilidades de crear objetos de forma textual? —los ojos de Antonia relampagueaban mientras con voz apenas contenida comenzaba a confiar a Cecilia una de sus frecuentes apologías—. Costó un poco de trabajo, eso sí; pero ahora podemos crear todo tipo de torres con unas pocas líneas de texto:

```
1 torre(altura=100,
2       radio=40,
3       espesor=5,
4       n pisos=10,
5       ladrillos_por_piso=20);
6 translate([100,0,0])
7 torre(altura=150,
8       radio=20,
9       espesor=10,
10      n pisos=50,
11      ladrillos_por_piso=20);
12 torre(altura=25,
13       radio=150,
14       espesor=8,
15       n pisos=5,
16       ladrillos_por_piso=20);
```

»Y no sólo eso: la posibilidad de crear tantas variantes nos incita, tan sólo viéndolas, a realizar mejoras y ampliaciones a lo ya conquistado: torres que no completen todo su perímetro

13.4. El texto completo, una apología y un poco de provocación

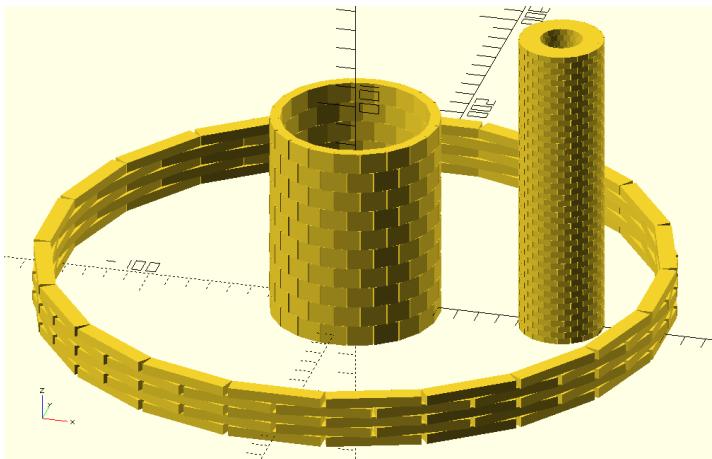


Figura 13.3: Torres creadas por Antonia.

(‘arcos’ de torres, diríamos), torres que vayan aguzándose hacia arriba, torres con almenas, torres con ladrillos faltantes... en fin: ¡todas las posibilidades que sólo las palabras logran sugerir a la imaginación!

Cecilia estaba casi convencida, pero rara vez podía acompañar a Antonia en esos vuelos pretenciosamente retóricos.

—Además, decime la verdad —el tono de Antonia se volvió súbitamente confidencial—: ¿a vos te parece que esta variedad, fiel a los delicados parámetros elegidos por el usuario, se puede lograr con *esos* programas que requieren el uso del ratón y los menús el 95 % del tiempo..?

Cecilia no podía sinceramente responder a esa pregunta; en todo caso, tal comparación le parecía impertinente y al borde de la provocación. Le pareció más productivo jugar un poco con sus torres:

```
1 for(r=[10:50:200])
2     torre(altura=200-r,
```

13. UNA TORRE MÁS O MENOS MEDIEVAL – III

```
3      radio=r,  
4      espesor=r*.2,  
5      n_pisos=(200-r)/10,  
6      ladrillos_por_piso=30);
```

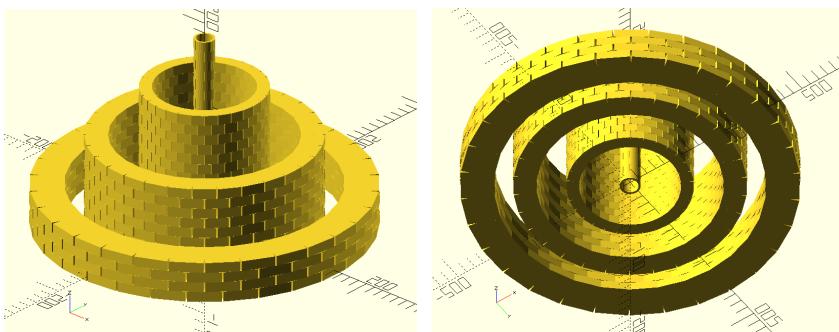


Figura 13.4: Torres anidadas con un bucle escritas por Cecilia.

— 14 —

El semicilindro

14.1. DIFERENCIAS

—*T*é acordás del reloj de Sol digital..? —preguntó al día siguiente Antonia, con aire inocente.

—Claro —Cecilia respondió con una sonrisa burlona—; pensé que nunca lo íbamos a empezar.

—Su cuerpo principal es un semicilindro; en él haremos luego los orificios que le den sentido y lo justifiquen. Así que te sugiero comenzar definiendo un nuevo objeto: el ‘semicilindro’ —propuso Antonia, y Cecilia, con tal de avanzar, asintió.

—En principio —continuó Antonia—, se me ocurre que un semicilindro debe quedar caracterizado por su altura y su radio; para no desentonar con el resto del lenguaje, y tomando en cuenta el estrecho parentezco entre un semicilindro y un ‘cylinder’, llamémoslos *h* y *r*:¹

```
1 module semicilindro(h,r){  
2 }
```

¹Esta decisión de Antonia debió haberse inspirado en el denominado “Principio de la mínima perplejidad” (*Principle of Least Astonishment*), que forma parte de la cultura informática desde la década de 1970, por lo menos. (Nota del Editor)

—Ahora bien —prosiguió Antonia—; podemos contemplar un semicilindro como un cilindro al que le cercenaron una mitad; por supuesto, esa operación elemental de ‘corte’ no podía faltar entre las posibilidades de OpenSCAD. Se logra con la operación `difference()`. Dejame mostrarte antes un ejemplo suelto:

```

1 $fn=100;
2
3 translate([-100,0,0])
4   cube([30,30,30],center=true);
5
6 translate([-50,0,0])
7   cylinder(h=60,r=5,center=true);
8
9 difference() {
10   cube([30,30,30],center=true);
11   cylinder(h=60,r=5,center=true);
12 }
```

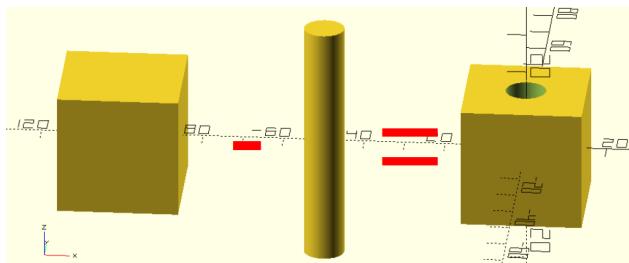


Figura 14.1: Antonia ofrece un ejemplo de la operación `difference()`.

»Las líneas 3 a 7 fueron para hacerme la canchera: sólo sirven para mostrar los elementos que luego resto efectivamente en las líneas 9 a 12. La idea es que `difference()` encierra entre llaves dos o más elementos, ‘restando’ del primero cada uno de los siguientes. A ver qué te parece este otro ejemplo:

```

1 $fn=100;
2 difference() {
3   cube([30,30,30],center=true);
4   cylinder(h=60,r=5,center=true);
5   rotate([90,0,0])
6   cylinder(h=60,r=5,center=true);
7   rotate([0,90,0])
8   cylinder(h=60,r=5,center=true);
9 }
```

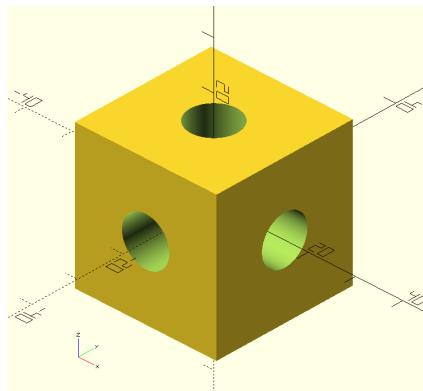


Figura 14.2: Tres cilindros perpendiculares restados a un cubo.

—A ver —Cecilia se entusiasmó y tomó el teclado—; dejame a mí:

```

1 $fn=100;
2 difference() {
3   cube([30,30,30],center=true);
4   for(vector=[[0,0,0],[90,0,0],[0,90,0]])
5     rotate(vector)
6     cylinder(h=60,r=5,center=true);
7 }
```

—¡Cómo te gustaron los bucles! —aprobó Antonia, y ambas rieron con ganas.

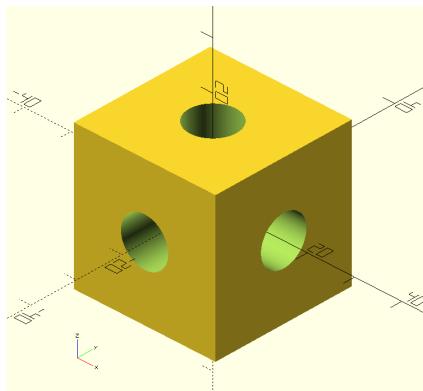


Figura 14.3: Tres cilindros perpendiculares restados a un cubo usando un bucle.

—Antonia —Cecilia preguntó—, ¿era necesario que los cilindros fueran más largos que el lado del cubo?

Antonia frunció los labios:

—Pues, sí. En caso contrario, no resulta claro qué debería pasar con las ‘tapas’: ¿deben permanecer (porque pertenecen al cuerpo inicial) o deben irse (porque pertenecen al cuerpo que se resta)? De hecho, fijate en la figura 14.4 lo que pasa si hago que los cilindros tengan la misma altura que el cubo:

```

1 $fn=100;
2 difference() {
3   cube([30,30,30],center=true);
4   for(eje=[[0,0,0],[90,0,0],[0,90,0]])
5     rotate(eje)
6     cylinder(h=30,r=5,center=true);
7 }
```

—A propósito —Antonia dedicó a Cecilia una mirada de admiración—: gracias a tu reescritura del texto, sólo tuve que cambiar el ‘60’ por el ‘30’ una sola vez.

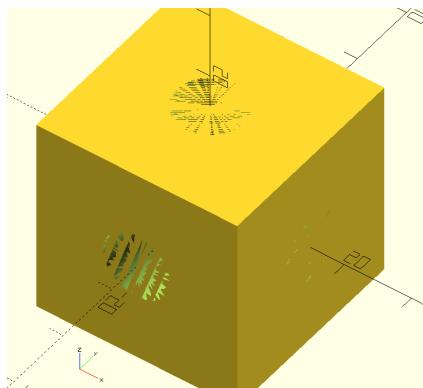


Figura 14.4: Los objetos sustraendo y minuendo no deben tener caras en común.

Cecilia se sonrojó ligeramente, y salió del paso con rapidez:

—Ahí lo veo, es verdad: el propio OpenSCAD pareciera indeciso acerca de qué hacer con esas ‘tapas’.

—Por esa razón conviene evitar toda ambigüedad haciendo los cuerpos ‘sustraendos’ más grandes que el cuerpo ‘minuendo’ —sentenció Antonia.

—¡La última! —Cecilia levantó bien alta la mano con una sonrisa—: ¿Por qué se escriben un par de paréntesis después de la palabra **difference**?

—Ni idea —fue la sincera y lacónica respuesta de Antonia.

14.2. MODIFICADORES

—¡Qué lastima que los cuerpos sustraendos desaparezcan! Quiero decir —Cecilia se explicó—, entiendo que si quiero restarlos, no deben permanecer; es obvio. Es sólo que siento que estoy escribiendo ‘a ciegas’: me gustaría poder ver de alguna manera los objetos que estoy restando.

—Hay una solución a esa razonable inquietud —reconoció Antonia—: todo cuerpo precedido por el símbolo '#' es dibujado en un tono rosado y transparente, sin importar si es sustraendo o no.

```

1 $fn=100;
2 difference() {
3     cube([30,30,30],center=true);
4     for(eje=[[0,0,0],[90,0,0],[0,90,0]])
5         #rotate(eje)
6         cylinder(h=30,r=5,center=true);
7 }
```

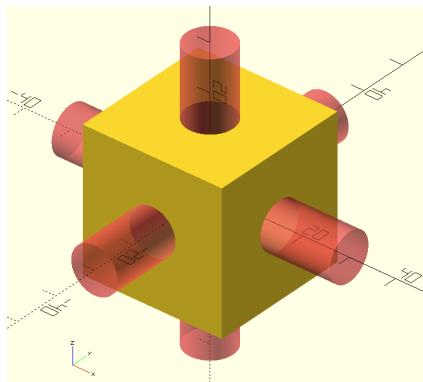


Figura 14.5: Antonia emplea el modificador visual "#".

—¡Hermoso! —exclamó Cecilia, contemplando la figura 14.5.

—Hay más modificadores —Antonia no ocultaba su regocijo cada vez que lograba despertar el entusiasmo en otra persona—. Uno que me resulta muy útil, particularmente cuando trabajo con un texto con muchos objetos, es '!': cuando precedés un objeto con él, OpenSCAD ignora todos los demás y lo dibuja a él solo. Te permite así concentrarte en un objeto en particular, sin distraerte con los demás —aclaró y, tras unos instantes en los que pareció

dudar, agregó—: A ver si lo encuentro...

Antonia estuvo un buen rato buscando en el desordenado árbol de directorios de su computadora hasta que dio finalmente con el par de imágenes de la figura 14.6.

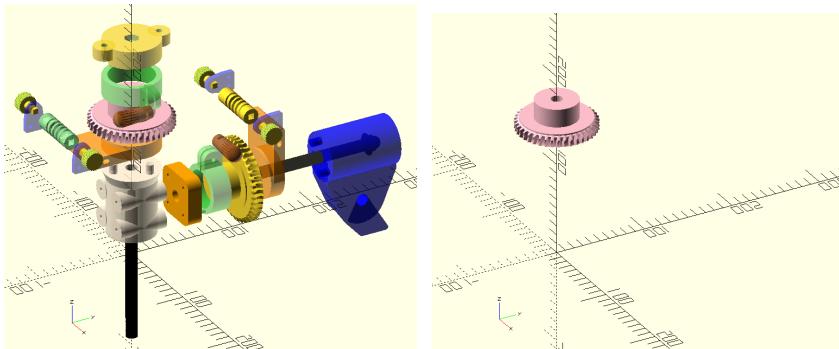


Figura 14.6: Una montura ecuatorial alemana. Gracias al modificador “!”, a la derecha sólo se muestra la corona del eje de declinación.

Cecilia casi saltó de su silla:

—¿¡Y eso qué es!? —preguntó en lo que fácilmente podía confundirse con un grito.

—Nada —respondió Antonia con gesto cansado—; una montura para un telescopio en la que vengo trabajando hace un tiempo con la esperanza de imprimirla algún día.²

²Asombrosamente, dicha montura no sólo fue terminada e impresa, sino incluso adosada a un telescopio actualmente en uso en el Observatorio donde desempeña sus tareas el autor. Debemos admitir también, no sin perplejidad, que hasta el momento no ha sido posible romperla ni malograrla; pero no dudamos de la labor eficaz del tiempo.

14.3. EL SEMICILINDRO (AHORA SÍ)

—Volvamos al semicilindro —invitó Antonia—. Creo que lo podemos resolver restando un cubo a un cilindro. Las dimensiones del cilindro ‘base’ deben ser las mismas que las del semicilindro deseado por el usuario, e indicadas por los parámetros h y r . Sólo hay que tener cuidado de que el cubo sustraendo sea más grande, y esté debidamente trasladado; a ver...

```

1 module semicilindro(h,r){
2     difference(){
3         cylinder(h=h,r=r);
4         #translate([-2*r,0,-h])
5         cube([4*r,2*r,3*h]);
6     }
7 }
8 semicilindro(40,5);

```

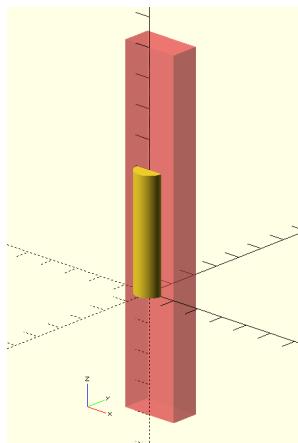


Figura 14.7: Módulo `semicilindro` en elaboración.

»Para ir viendo cómo queda, y hasta que quedemos conformes con la definición final, yo diría que usemos el modificador ‘#’ para

el cubo —propuso Antonia.

—¡Eh! —objetó Cecilia—. ¿No es demasiado grande el cubo?

—‘Demasiado’ para qué? —se defendió Antonia—. Ese cubo no ocupa más lugar en la memoria de la computadora que uno visualmente más pequeño, y nos permite mantener el texto con números más manejables y discretos que 1,75, 2,5, u otros por el estilo.

Cecilia, tras considerarlo unos instantes, sintió que su amiga tenía razón. En todo caso, tampoco parecía digno de discusión.

Antonia recorría el texto con una mirada que trasuntaba una evidente insatisfacción:

—Hay algo que podemos mejorar —lanzó finalmente—. Como sin duda recordarás, los cilindros admiten, al momento de ser creados, la indicación `center=true`; considero que, a fin de cumplir con el ‘Principio de mínima perplejidad’³, los semicilindros deberían también hacerlo:

```

1 module semicilindro(h,r,center=false){
2     difference(){
3         cylinder(h=h,r=r,center=center);
4         #translate([-2*r,0,-h])
5         cube([4*r,2*r,3*h]);
6     }
7 }
8
9 semicilindro(40,5);
10
11 translate([100,0,0])
12 semicilindro(40,5,center=true);
```

Cecilia tuvo que repasar mentalmente sus recuerdos de los parámetros opcionales. En principio, entendió que el semicilindro aceptaba un parámetro `center`, con un valor preestablecido (o

³¡Ah! ¿Vieron? Antonia conocía ese principio, después de todo. (Nota del Editor)

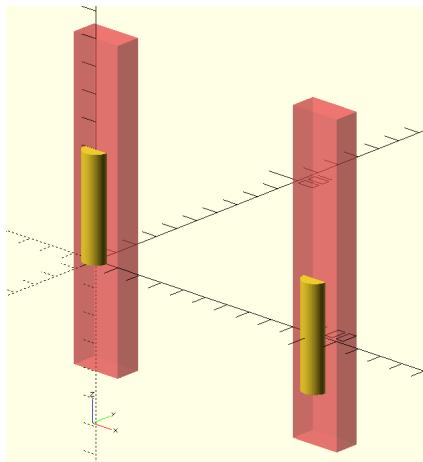


Figura 14.8: El módulo `semicilindro` acepta el parámetro `center`.

“por defecto”, como les gusta traducir a algunos informáticos⁴) igual a `false`; es decir que, si el usuario no lo menciona (como ocurría en la línea 9 del nuevo texto), adoptaba ese valor. En caso contrario (como en la línea 12), puede tomar el valor `true`. En cualquiera de ambos casos, el módulo `semicilindro` pasa ese valor al parámetro `center` del `cylinder`: o sea, si el usuario escribe “`center=true`”, el `cylinder` se crea con `center=true`; en caso contrario, con `center=false`. A Cecilia le pareció una solución muy limpia: un “pase de manos”, por así decir. El trabajo de centrar el semicilindro resultaba finalmente responsabilidad del cilindro base, del cual podemos confiar que sabe muy bien cómo centrarse solo.

⁴Y hacen bien: la vigesimotercera edición del diccionario de la Real Academia admite la locución adverbial “*por defecto*” en el sentido en que se emplea comúnmente en Informática y Tecnología. (Nota del Editor)

14.4. COLORES

—Yo creo que la definición quedó bastante bien —propuso Antonia, guiñando un ojo—. ¡Vamos sacar el modificador '#' y a usar un poco nuestros semicilindros!

```

1 module barra_de_chocolate(){
2   for (x=[0:10:100])
3     translate([x,0,0])
4     rotate([-90,0,0])
5     semicilindro(h=50,r=5);
6 }
7
8 color("SaddleBrown")
9 barra_de_chocolate();

```

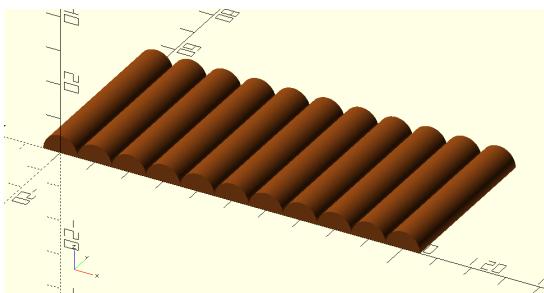


Figura 14.9: Una barra de chocolate... o algo así.

—Muy graciosa —soltó Cecilia, que no siempre compartía el sentido del humor de Antonia—. ¿Qué es eso de `color`?

—Es una transformación más, como `rotate` y `translate`; sólo que afecta meramente al color de la visualización —Antonia sonreía, divertida—. Puede usarse con el nombre del color (de acuerdo a una prolífica lista que puede consultarse en https://en.wikibooks.org/wiki/OpenSCAD_User_Manual/Transformations#color), o usando el clásico sistema de código RGB. Es posible,

además, indicar un factor de transparencia... en fin, los detalles te sugiero consultarlos en la página cuya dirección acabo de copiarte.

Cecilia comprendió que Antonia no estuviera demasiado ansiosa por dar más detalles: ya iban por la página 12 del capítulo, que fue bastante intenso.

— 15 —

Divertimento: un bongo

CUANDO CECILIA entró a la oficina de Antonia encontró una sorpresa sobre su escritorio.
—¿Y eso..? —preguntó.



Figura 15.1: Un bongo.

—Un bongo —respondió Antonia con naturalidad.

—¿Un bongo? —preguntó nuevamente Cecilia.

—Sí, un bongo —reafirmó su amiga con cara de sorpresa—. Los bongos eran personajes del juego '*Dr. Peppers and the bongos*'. No me digas que no te acordás...

Cecilia arqueó las cejas demostrando ignorar la existencia de ese juego.

—Ay, Cecilia; ¡por favor! —Antonia subrayó su asombro separando los brazos del cuerpo ostensiblemente—. ¿Dónde estuviste en la década del '80? *Dr. Peppers and the bongos* fue un juego revolucionario para su época: exhibió el primer motor 3D de la historia. Era un tanto rudimentario, es verdad; pero muy jugable. Además de que como juego era muy divertido.¹ En fin, ¿te gusta?

Cecilia miró de cerca el bongo. Le pareció bonito y simpático.

—Sí —contestó.

—Qué bueno, porque es tuyo —la mirada de Cecilia debió haber reflejado su sorpresa, porque Antonia sintió la necesidad de ser más clara—: Sí; es un regalo.

Cecilia no sabía qué decir: era la primera vez que Antonia le hacía uno.

—Gracias... —pudo soltar apenas, tras unos instantes.

15.1. HULL()

Cecilia movía suavemente el bongo entre sus manos:

—Antonia, pensé que OpenSCAD sólo permitía crear objetos más bien... 'geométricos'. ¿Como obtuviste estas formas tan redondeadas y suaves? —quiso decir "tiernas", pero no se animó.

¹ Admito que la referencia de Antonia al juego *Dr. Peppers and the bongos* me toma por sorpresa a mí también. He fatigado Internet en busca de mayor información y no he podido arribar a nada concreto, salvo unos enlaces rotos a una efímera página de Wikipedia que al parecer fuera borrada por manos anónimas. (Nota del Editor)

—Tengo mis trucos —deslizó Antonia con un guiño—. El primero es la transformación `hull()`:

```
1 $fn=100;
2 hull() {
3   sphere(r=10);
4   translate([0,0,30])
5   sphere(r=5);
6 }
```

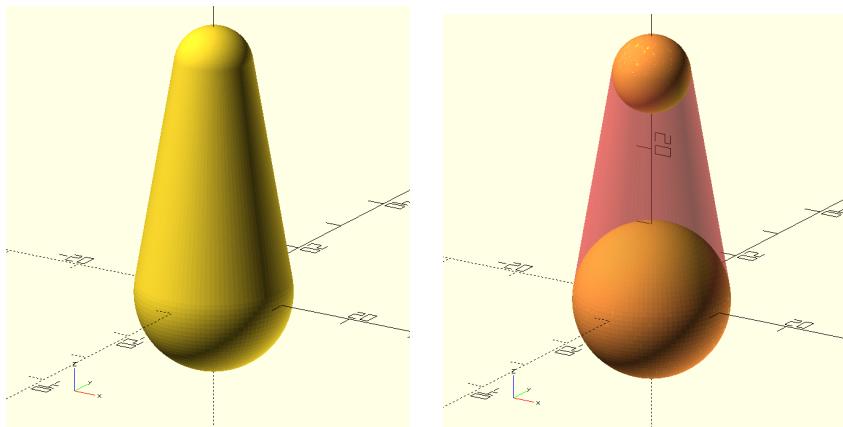


Figura 15.2: Antonia ofrece un rápido ejemplo de la operación `hull()`.

»`hull()` toma los objetos que le pasás entre llaves y crea un nuevo objeto que es su envolvente convexa —Antonia miró fugazmente el techo como buscando algo más parecido a una explicación—. Imaginate que rodeas a los objetos entre llaves con una suerte de envoltura de goma bien ajustada; bueno, eso es lo que hace `hull()`. En la imagen izquierda de la figura 15.2 te muestro el resultado del texto que escribí; a la derecha destaqueé artificialmente las dos esferas entre llaves, para que se note mejor su estructura.

Cecilia miraba el monitor con una cara que no inspiraba confianza en los poderes explicativos de Antonia.

—A ver con otro ejemplo... —insistió—. Ahora voy a envolver un cubo en la base con un cilindro en la punta:

```

1 $fn=100;
2 hull() {
3   cube([30,30,10],center=true);
4   translate([0,0,40])
5   cylinder(h=5,r=5);
6 }
```

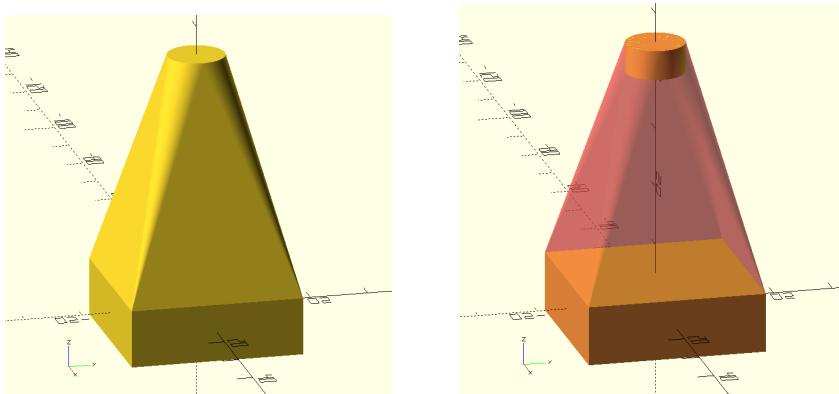


Figura 15.3: Otro ejemplo de `hull()`.

»Acordate de que el objeto producido por `hull()` es el de la izquierda; el otro lo hice para destacar su estructura interna, nomás —alertó Antonia, y continuó—: Con esto, ya podemos escribir el cuerpo del bongo.

```

1 $fn=100;
2 module cuerpo(){
3   hull() {
4     translate([0,0,3]) sphere (d=12);
5     translate([0,0,11.25]) sphere (d=6);
```

```
6     }  
7 }  
8 cuerpo();
```

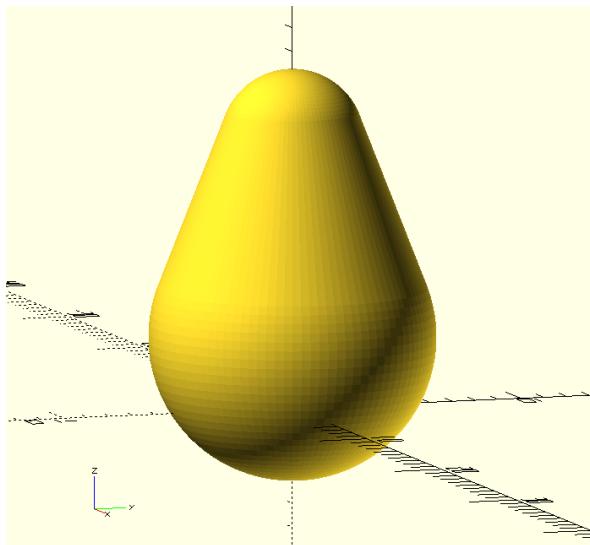


Figura 15.4: Cuerpo del bongo.

Cecilia sonrió, no sabemos si porque comprendía cabalmente lo que Antonia estaba haciendo o porque el cuerpo del bongo había quedado muy lindo.

—¿Por qué usaste esos valores? En particular el 11,25? —preguntó.

—Ay, Cecilia... ¡porque sí! —respondió su amiga con una risa ligera—. Dale, relajate... no todo tiene que depender de cálculos estrechos y rigurosos... pongamos números ‘a ojito’ y ya fue...

Cecilia estaba sorprendida; nunca había visto a Antonia con ese desenfado. Pero inmediatamente recordó que desde siempre parecía reservarse el privilegio de desconcertarla. Y el bongo había quedado realmente bonito, así que decidió seguirle el juego.

—Fijate —explicó Antonia—: usé dos esferas y las envolví en un `hull()`. En la figura 15.5 te destaco, una vez más, la estructura interna.

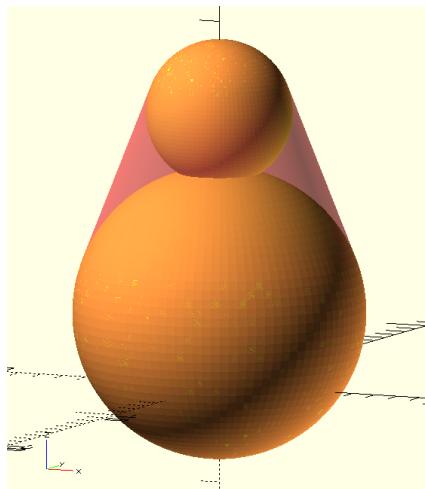


Figura 15.5: Estructura del cuerpo del bongo, rodeada por el `hull()`.

15.2. SCALE()

»Ahora creo que es el turno de la cabeza. Como sabemos, los bongos tienen una cabeza más ancha que alta. Para lograrla usaremos la transformación `scale()`.

»Como su nombre sugiere, `scale()` realiza un escalado del objeto al cual se aplica en cada uno de sus tres ejes. En el caso de la cabeza del bongo, como podés ver en la figura 15.6, toma la esfera y deja igual su tamaño en X e Y, pero multiplica su longitud en Z por 0,8. Debido a eso, la ‘aplasta’ —Antonia replicó el efecto acercando las palmas de sus manos dispuestas horizontalmente,

```

1 module cabeza() {
2     translate([0,0,12.75])
3     scale([1,1,.8])
4     sphere(d=9);
5 }
6
7 cabeza();

```

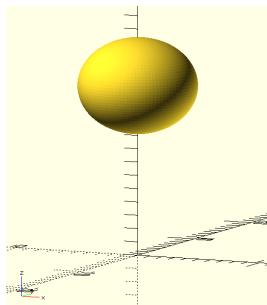


Figura 15.6: Antonia usa la transformación `scale` para dar forma a la cabeza del bongo.

una encima de la otra; al parecer, pertenecía al grupo de docentes que consideran que mover las manos equivale a explicar—. El escalado te permite, por ejemplo, realizar cilindros de base ovalada, o ‘pastillas’; fíjate la figura 15.7.

```

1 scale([1,.5,1])
2 cylinder(r=10,h=30);
3
4 translate([40,0,15])
5 scale([2,1,.5])
6 sphere(r=10);

```

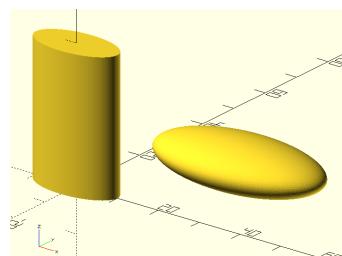


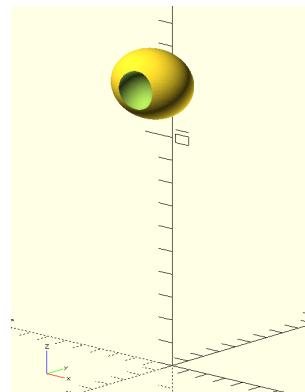
Figura 15.7: Antonia ofrece dos ejemplos más de la transformación `scale`.

»La ‘pastilla’, como verás, fue alargada dos veces en el eje X y achicada a la mitad en el Z —desarrolló Antonia, y agregó—: El escalado me resultó útil para escribir también el ojo del bongo.

```

1  module ojo() {
2    translate([1.8, -3.25, 13.35]) {
3      difference() {
4        // globo ocular
5        scale([1.3, 1, 1])
6        sphere(d=3);
7        // iris
8        translate([0.25, -1.56, 0])
9        sphere(d=1.8);
10      }
11    }
12  }
13 ojo();

```

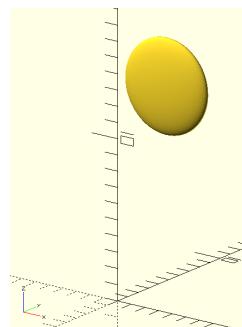


»Para terminar con la cabeza nos faltan las orejas que caracterizaron, desde tiempos inmemoriales, a los bongos —Antonia tecleó con rapidez, mientras ladeaba ligeramente la cabeza como quien comprueba el efecto de las pinceladas sobre un lienzo:

```

1  module oreja() {
2    translate([3.6, 0, 13.95])
3    scale([1, .2, 1])
4    sphere(d=6);
5  }
6 oreja();

```



—A ver cómo está quedando... —intervino Cecilia, que ya tenía ganas de escribir.

—¡Qué lindo! —exclamó Cecilia ante la figura 15.8—. Encima parte de la cabeza sobresale del iris y queda como una pupila... Voy a repetir ojo y oreja trasladándolos la distancia adecuada.

```

1  cuerpo();
2  cabeza();
3  ojo();
4  oreja();

```

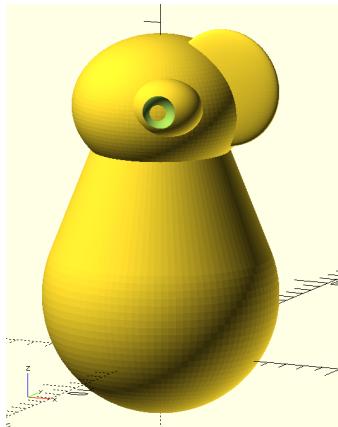


Figura 15.8: Bongo a medias.

Antonia la detuvo, guiñando un ojo:

—Hay una manera mejor de repetir objetos simétricamente; te sugiero que sigamos con el brazo y la pierna izquierdos, y después te cuento cómo replicar todo del lado derecho del bongo.

A Cecilia le pareció natural que OpenSCAD contara con una transformación que diera cuenta de los modelos que presentan simetría bilateral, así que decidió confiar en Antonia.

—El resto de los miembros del bongo no representa ninguna herramienta nueva; se trata de escribir, nomás —adelantó Antonia.

```

1  module brazo () {
2    rotate([0,0,-20])
3    translate([4.2,0,9])
4    rotate([20,159,0]) {
5      // brazo
6      hull () {
7        sphere(d=3);
8        translate([0,0,6])
9          sphere (d=2.4);
10     }
}

```

15. DIVERTIMENTO: UN BONGO

```
11      // mano
12      translate([-0.12,0,6])
13      scale([.69,1,1])
14      sphere (d=3.6);
15  }
16 }
17 brazo();
```

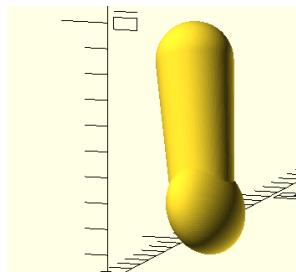


Figura 15.9: Brazo del Bongo

```
1 module pierna(){
2     translate([4.2,0,0])
3     rotate([90,0,0]){
4         // pierna
5         hull(){
6             sphere(d=6);
7             translate([0,0,8.4])
8             sphere(d=4.8);
9         }
10        // pie
11        hull(){
12            translate([0,0,9.6])
13            scale([1,1,.6])
14            sphere(d=6);
15            translate([0,3.6,9.6])
16            scale([1,1,.4])
17            sphere(d=3.6);
18        }
```

```

19      }
20  }
21  pierna();

```

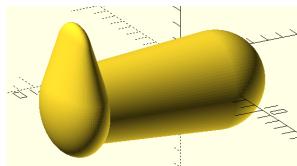


Figura 15.10: Pierna del Bongo

»Vamos a ver todo junto —propuso Antonia, mientras producía la figura 15.11.

```

1  cuerpo();
2  cabeza();
3  ojo();
4  oreja();
5  brazo();
6  pierna();

```

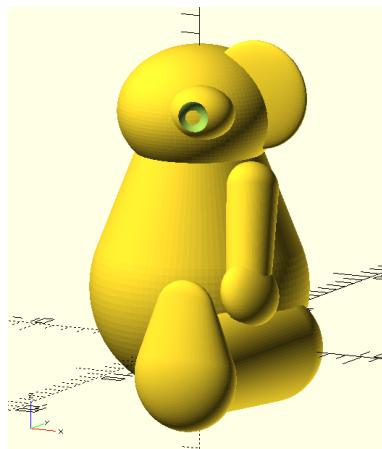


Figura 15.11: Medio bongo.

15.3. MIRROR()

—Lindo, ¿no? —Antonia buscó la aprobación de su amiga—. Ahora debemos ‘espejar’ ojo, oreja, brazo y pierna. Pues bien,

resulta que OpenSCAD cuenta con una transformación que justamente realiza esa operación —añadió, mientras escribía el ejemplo de la figura 15.12.

```
1  cuerpo();  
2  cabeza();  
3  ojo();  
4  oreja();  
5  brazo();  
6  pierna();  
7  mirror([1,0,0]){  
8    ojo();  
9    oreja();  
10   brazo();  
11   pierna();  
12 }
```

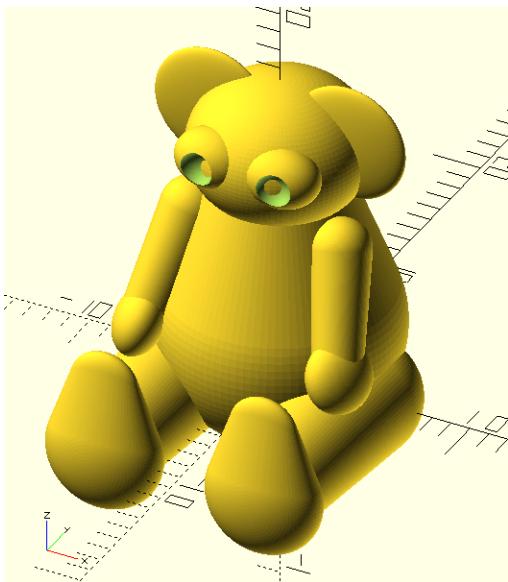


Figura 15.12: Antonia replica ojo, oreja, brazo y pierna del bongo mediante la transformación `mirror()`.

»`mirror()` copia especularmente (y espectacularmente) —Antonia era famosa en todo Harvard por sus chistes malos y ñoños— los objetos a los que se aplica (si son más de uno, deben colocarse entre llaves, como con cualquier otra transformación). Para decidir cuál es el plano del ‘espejo’, debemos indicar entre paréntesis un vector cualquiera que sea perpendicular al mismo; mirá el ejemplo de la figura 15.13.

»En las líneas 2 a 4 escribí una esfera roja, que luego copié en las líneas 7 a 9, salvo que de color azul —explicó Antonia, aunque Cecilia era muy capaz de verlo por sí misma—. En la línea

```

1 // esfera original
2 translate([10,0,0])
3 color("red")
4 sphere(r=1);
5 // espeja espejada
6 mirror([6,0,0])
7 translate([10,0,0])
8 color("blue")
9 sphere(r=1);

```

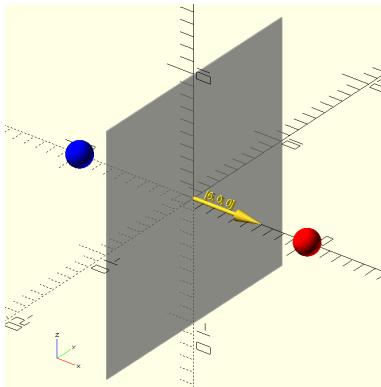


Figura 15.13: Antonia ofrece un ejemplo de la transformación `mirror()`.

6 antepuse a la segunda esfera la transformación `mirror([6,0,0])`. La función del vector `[6,0,0]` es definir el plano que indiqué en gris (artificialmente, sólo a fines de que se notara). Y este plano es el que funciona de ‘espejo’ para reflejar la esfera y ubicarla donde está, en lugar de superponerla a la roja.

—¿Y por qué aparece el vector? —cuestionó Cecilia.

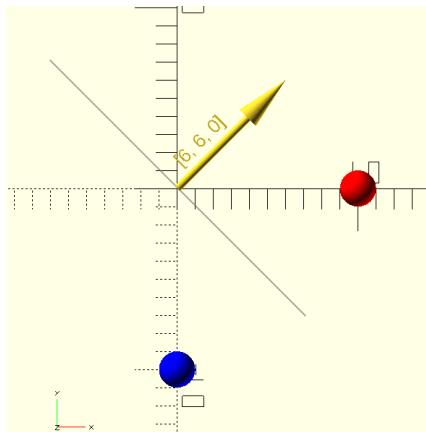
—¡Ah! Es verdad; también lo coloqué artificialmente, para que se notara. `mirror()` no dibuja el vector —agregó Antonia—. La magnitud del vector tampoco es importante, sino sólo su dirección: el resultado sería exactamente el mismo si hubiéramos usado `mirror([1,0,0])`; el problema sería que mi vector artificial y pictórico sería diminuto y no se vería. El vector puede apuntar en cualquier dirección, por supuesto; el plano ‘reflectante’ será siempre el perpendicular al mismo.

Antonia escribió otro ejemplo para Cecilia, quien no estaba segura de entender del todo. Se propuso escribir luego otros ejemplos por sí misma para terminar de aprehender esta nueva transformación.

```

1 // esfera original
2 translate([10,0,0])
3 color("red")
4 sphere(r=1);
5 // espeja espejada
6 mirror([6,6,0])
7 translate([10,0,0])
8 color("blue")
9 sphere(r=1);

```

Figura 15.14: Otro ejemplo de `mirror()`.

15.4. UNION()

—Ahora bien, para imprimir el bongo debemos darle una base plana. A mí me resultó natural restar un cubo debajo —propuso Antonia.

```

1 module bongo(){
2   difference(){
3     union(){
4       cuerpo();
5       cabeza();
6       brazo();
7       pierna();
8       ojo();
9       oreja();
10      mirror([1,0,0]){
11        brazo();
12        pierna();
13        ojo();
14        oreja();
15      }

```

```

16      }
17      translate([0,0,-7.7])
18      cube ([24,24,12],center=true);
19  }
20 }
21 bongo();

```

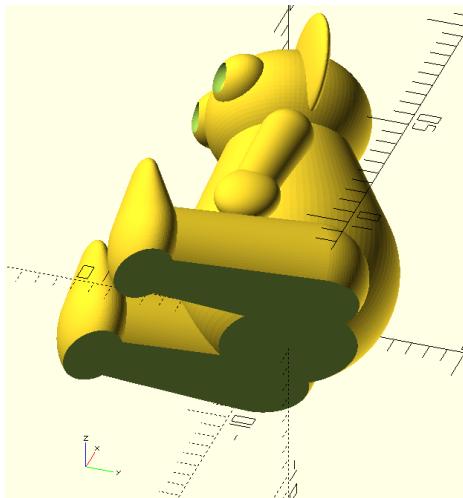


Figura 15.15: Base del bongo.

»Para ello tuve que usar la transformación `union()`. Bueno, en realidad, no ‘transforma’ nada: une los objetos que recibe entre llaves como uno solo. Debí usarlo porque yo quiero que `difference()` reste el cubo de las líneas 17 y 18 a *todo* lo anterior. Es decir, desde el punto de vista de `difference()`, el objeto minuendo es el que resulta de `union()`, que dentro de sí abarca a los objetos de las líneas 4 a 14, y el único cuerpo sustraendo resulta ser el cubo final.

Ambas amigas contemplaban satisfechas el monitor, cuando Antonia se irguió de golpe en su silla, quizás sobrereactuando un

poco la alarma:

—¡Oh, no! ¡Por poco me olvido! Un bongo no es un bongo sin rabito:

```
1 module rabito(){
2     translate([0,4.8,-0.3])
3     scale([1.2,1.2,1.2])
4     sphere(d=3);
5 }
6 bongo();
```

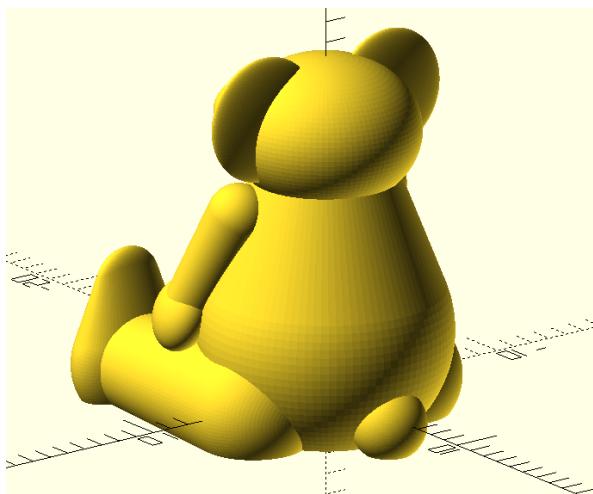


Figura 15.16: Bongo completo, con todo y rabito.

15.5. MÓDULOS DENTRO DE MÓDULOS

Antonia aún no parecía satisfecha del todo:

—Supongamos que quisiéramos crear otro muñeco. Es razonable pensar que debería tener sus propios ojos, cabeza, brazos, piernas, orejas y, tal vez, incluso un rabito. Pero en ese caso debe-

ríamos bautizar sus módulos con nombres que no colisionen con los del bongo: algo como `cabeza_vaquita`, por ejemplo.

Cecilia no veía ningún problema en ello, pero tampoco creyó oportuno disentir al menos hasta ver adónde quería llegar su amiga.

—Creo que los módulos que definen los miembros de los respectivos muñecos deberían pertenecer unívocamente a los mismos —argumentó Antonia—. Después de todo, ¿dónde más se puede usar la oreja de un bongo que no sea en la cabeza de un bongo..?

»Por lo tanto, lo que podemos hacer es aprovechar la posibilidad que nos brinda OpenSCAD de escribir módulos dentro de módulos:

```
1 module bongo(){
2     module cuerpo(){
3         [...]
4     }
5     module cabeza(){
6         [...]
7     }
8     module brazo(){
9         [...]
10    }
11    [...]
12    difference(){
13        union(){
14            cuerpo();
15            cabeza();
16            [...]
17        }
18        translate([0,0,-7.7])
19        cube ([24,24,12],center=true);
20    }
21 }
```

»De esta forma, los módulos `cuerpo`, `cabeza`, etc., no pueden ser invocados ni aludidos desde fuera del módulo `bongo`, por lo

que ningún conflicto causarán con otras extremidades de otros eventuales muñecos.

Cecilia no pudo negar que, al menos como posibilidad, resultaba interesante.

La idea del reloj, un poco más precisa

—B^{IEN}, CECILIA —comenzó Antonia, con gesto adusto—; ahorra sí la cosas empezarán a complicarse.

—¡Por fin! —exclamó Cecilia, simulando una confianza en sí misma que no estaba tan segura de sentir.

—Antes de sumergirnos definitivamente en nuestro reloj, aún debemos conquistar unas pocas herramientas más de OpenSCAD; pero creo que éste es un buen momento para justificarlas previamente precisando un poco más nuestro objetivo —Antonia desgranaba su explicación con lentitud; Cecilia temió que estuviera por caer en una de sus tantas procrastinaciones: cuando estaba a punto de terminar un proyecto, empezaba a dar vueltas y más vueltas, como si concluir algo (cualquier cosa que fuere) supusiera una pérdida o una despedida que quisiera evitar o demorar—. El cuerpo principal de nuestro reloj, como dijimos, será un semicilindro:

```
1 module cuerpo(largo ,  
2                 radio ,  
3                 center=false){  
4     rotate([-90,0,0])  
5     semicilindro(h=largo ,  
6                   r=radio ,  
7                   center=center ,
```

```
8           $fn=200) ;  
9 }  
10 cuerpo(150,30,center=true);
```

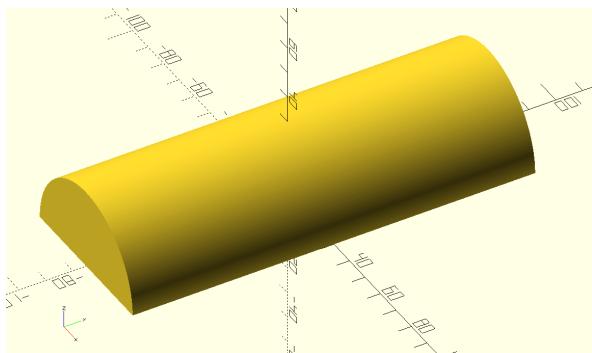


Figura 16.1: El cuerpo del reloj.

»Luego revisaremos las medidas —aclaró—; usé esos valores sólo para fijar ideas.

—¡Pasaste el \$fn=200 como parámetro del semicilindro en la línea 8! —Cecilia estaba muy atenta.

—Ah, sí; no te conté —reconoció Antonia—; es una posibilidad muy interesante: si hacés eso, la variable \$fn sólo aplica al cuerpo que la recibe. De esa forma, no obligás a todos tus objetos a respetar la misma configuración.

»Ahora bien —Antonia avanzó a tientas, como quien busca la mejor forma de una idea—; supongamos que podemos determinar la dirección del Sol en un momento dado del día... y lo seremos, por supuesto: ¡al final, somos astrónomas! —sonrió, y Cecilia le devolvió la sonrisa, alentándola a continuar—. Entonces, deberemos cortar el cuerpo principal con unos paralelepípedos orientados según esa dirección, como los de las figuras 16.2.

Los ojos de Cecilia parecían querer comerse el monitor. Lo que allí pasaba era demasiado hermoso para ser verdad... y también

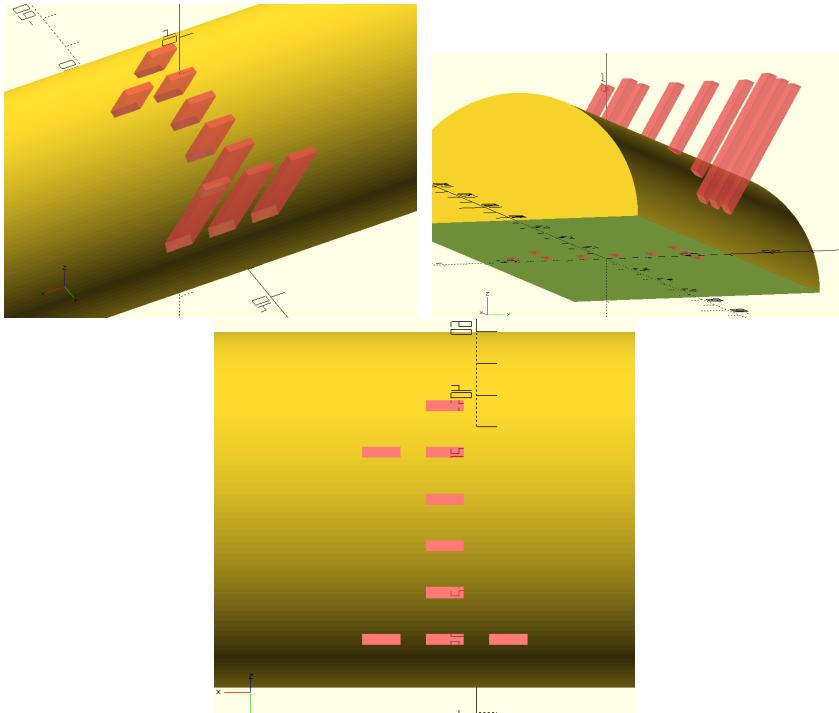


Figura 16.2: Palalelepípedos con los que Antonia propone hora-
dar el cuerpo del reloj, a fin de formar los dígitos que señalarán
la hora. Podemos considerar al Sol tan lejos de la Tierra que sus
rayos llegarán al reloj de manera paralela entre sí.

demasiado arduo. De pronto, en un vértigo, sintió que no podría repetir, ni aún siquiera comprender, lo que allí estaba ocurriendo. Instintivamente giró sobre sí misma para mirar a Antonia: esperaba encontrar en ella la seguridad que le faltaba, pero sorprendió en sus ojos un reflejo de su propia e incipiente angustia. Comprendió —conociéndola como la conocía— que ella también estaba preocupaba: el fantasma de ser una impostora como docente siempre la acechaba. Así que ahí estaban las dos: angustiada

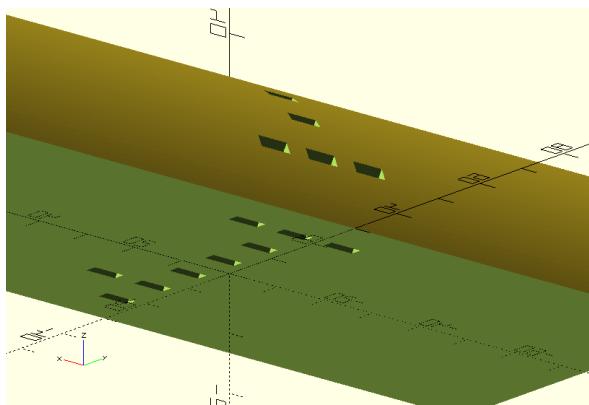


Figura 16.3: Agujeros que forman el dígito 1.

una por poder entender, y la otra por saber explicar. Cecilia supo entonces que la única salida que tenían debía recorrerse juntas: aliadas frente al mismo problema.

Antonia no pudo seguir, naturalmente, el pensamiento de su amiga, pero algo debió intuir cuando finalmente la vio sonreír, porque inmediatamente su rostro se despejó y los rayos de una tímida sonrisa destellaron en él.

—Si borro efectivamente los paralelepípedos quedaría como en la figura 16.3 —dijo.

—Hermoso —pronunció esta vez Cecilia en voz alta, con la seguridad de que juntas podrían hacer no sólo un reloj, sino un almanaque entero si hacía falta.

Polígonos y extrusiones

17.1. POLÍGONOS

SUPONGO QUE uniendo y cortando cubos, cilindros y esferas podrían lograrse todos los objetos imaginables; he ahí un bonito teorema para demostrar —conjeturó Antonia—. En cualquier caso, es bueno saber que OpenSCAD nos brinda otras posibilidades: por ejemplo, crear una superficie bidimensional para luego extenderla perpendicularmente. Una de esas superficies es el [polygon](#).

```
1 polygon([[-20,-20],  
2           [-10,0],  
3           [-20,20],  
4           [20,20],  
5           [10,0],  
6           [20,-20]]);
```

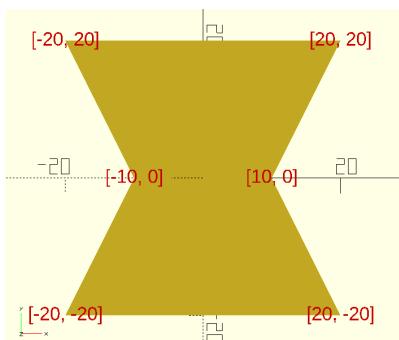


Figura 17.1: Antonia crea un [polygon](#) a manera de ejemplo.

»`polygon` recibe como parámetro una matriz con los puntos que lo definen —explicó Antonia—. Cada uno de ellos representa un punto en el plano XY: el primer número del punto es la coordenada en X, y el otro la posición en Y; por supuesto, al tratarse de una figura 2D dichos puntos no tienen coordenada Z. OpenSCAD los une entonces, en orden, mediante segmentos. El último punto se une automáticamente con el primero: en el caso de la figura 17.1, el punto [20, -20] con el [-20, -20].

»Si vos mirás el polígono de soslayo —advirtió Antonia— te va a aparecer con un pequeño espesor; pero que no te confunda: es una característica del motor gráfico que emplea OpenSCAD para visualizar los objetos. Los polígonos, internamente y de acuerdo con la más elemental geometría, no tienen espesor. De paso, te confirmo que los puntos pueden aludirse con una variable.

```
1  puntos=[[-20,-20],  
2      [-10,0],  
3      [-20,20],  
4      [20,20],  
5      [10,0],  
6      [20,-20]];  
7  
8  polygon(puntos);
```

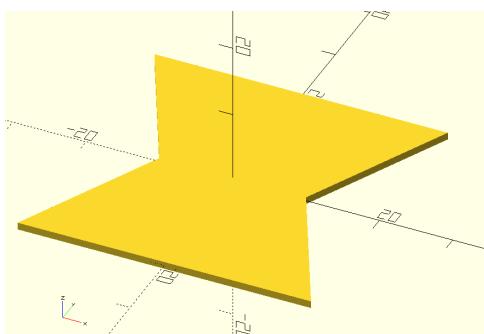


Figura 17.2: `polygon` definido a partir de una variable. El espesor que se aprecia es un artificio del motor gráfico de OpenSCAD: internamente y como la más elemental geometría impone, es nulo.

17.2. EXTRUSIÓN LINEAL

—Hay varias operaciones que te permiten erigir un objeto 3D con una figura plana; quizá la más elemental sea la ofrecida por `linear_extrude` —dijo Antonia, mientras tecleaba con su rapidez habitual.

```

1  puntos=[[ -20 , -20] ,
2      [-10 ,0] ,
3      [-20 ,20] ,
4      [20 ,20] ,
5      [10 ,0] ,
6      [20 , -20]] ;
7
8  linear_extrude(100)
9  polygon(puntos);

```

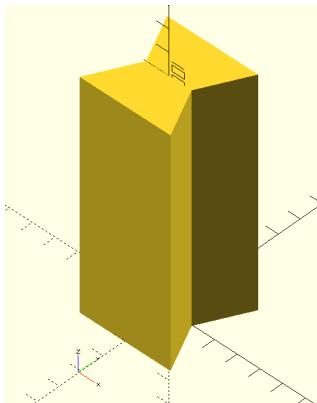


Figura 17.3: `polygon` extrudido mediante la operación `linear_extrude`.

»El primer parámetro de `linear_extrude` indica la distancia que la figura debe ser extrudida¹ a lo largo del eje Z —aclaró Antonia.

TWIST

»Hay más posibilidades —Antonia pareció entusiasmarse—: con el parámetro `twist` podés hacer girar la figura mientras la extrudís.

¹Confieso que la conjugación del verbo extrudir me toma por sorpresa: decidí confiar en los servicios de la página <https://dle.rae.es/extrudir>.

```
1  puntos = [[-20,-20],  
2      [-10,0],  
3      [-20,20],  
4      [20,20],  
5      [10,0],  
6      [20,-20]];  
7  
8  linear_extrude(100, twist=90)  
9  polygon(puntos);
```

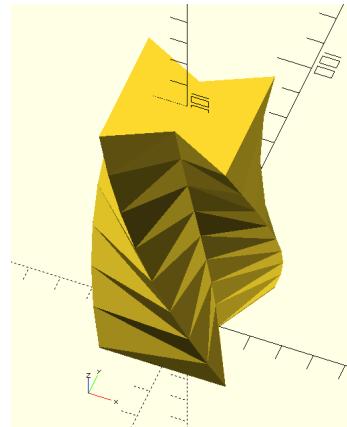


Figura 17.4: `polygon` extrudido y girado gracias a la opción `twist` de `linear_extrude`.

»El valor asignado a `twist` representa el giro total a lo largo de la extrusión. En el ejemplo que escribí, el polígono giró 90° entre las posiciones inicial y final.

SLICES

Antonia miraba el resultado con cierta insatisfacción:

—Por defecto, OpenSCAD genera 20 ‘rodajas’ cuando gira una figura durante una extrusión; por eso esa torre helicoidal nos quedó medio... tosca. Pero podemos hacerla más suave (o sea, con más ‘rebanadas’) aprovechando el parámetro `slices`.

»Quizá se me fue la mano con el valor 200 —Antonia rio ligeramente contemplando la figura 17.5—; pero la idea es ésa.

SCALE

»Hay más, pero te molesto con la última —rogó Antonia:

```

1  puntos = [[-20,-20],
2      [-10,0],
3      [-20,20],
4      [20,20],
5      [10,0],
6      [20,-20]];
7
8  linear_extrude(100, twist=90,
9      slices=200)
10 polygon(puntos);

```

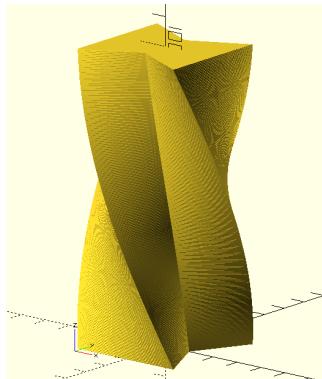


Figura 17.5: El `polygon` resulta con un contorno más suave gracias a la opción `slices` de `linear_extrude`.

```

1  puntos = [[-20,-20],
2      [-10,0],
3      [-20,20],
4      [20,20],
5      [10,0],
6      [20,-20]];
7
8  linear_extrude(100, twist=90,
9      slices=200, scale=0)
10 polygon(puntos);

```

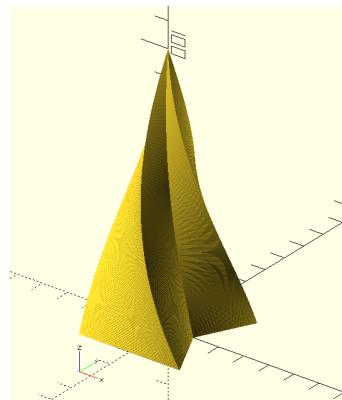


Figura 17.6: El `polygon` termina su extrusión en punta debido al uso de la opción `scale=0` de `linear_extrude`.

»Con el parámetro `scale` podés determinar el tamaño de la figura en su posición final con respecto a la inicial.

»Con `scale=0` la figura final se anula, por lo que la extrusión termina en punta —explicó Antonia, señalando la figura 17.6—.

También podés, por supuesto, hacer que la tapa sea más grande que la base, como podés apreciar en la figura 17.7.

```

1  puntos = [[-20,-20],
2      [-10,0],
3      [-20,20],
4      [20,20],
5      [10,0],
6      [20,-20]];
7
8  linear_extrude(100, twist=90,
9      slices=200, scale=3)
10 polygon(puntos);

```

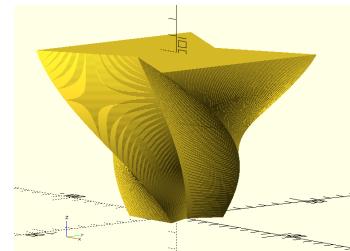


Figura 17.7: `linear_extrude` con la opción `scale=3`.

Cecilia estaba fascinada con las posibilidades, pero aún no podía ver qué relación mantenían con el reloj solar ansiado.

17.3. EXTRUSIÓN ROTATORIA

—Hay una extrusión más que te quiero mostrar —Antonia estaba visiblemente entusiasmada—. No la vamos a usar en el reloj pero, ¿qué más da? Está muy buena.

```

1 $fn=200;
2
3 rotate_extrude(angle=360)
4     translate([100,0,0])
5     polygon(puntos);

```

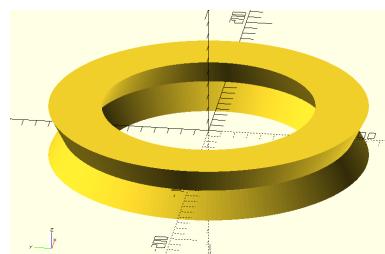


Figura 17.8: Antonia ofrece un ejemplo de `rotate_extrude`.

Antonia pareció buscar la mejor manera de explicar el nuevo giro lingüístico:

—`rotate_extrude` genera un sólido rotando alrededor del eje Z un polígono cualquiera; en el ejemplo de la figura 17.8, usé el que ya definimos antes. Es importante que el polígono íntegro se ubique de un mismo lado del eje Y —advirtió—; por esa razón, tuve que trasladarlo primero. En caso contrario, OpenSCAD se hubiera quejado con un mensaje de error.

Cecilia no entendía un detalle:

—Antonia, el polígono lo creamos recostado sobre el plano XY; ¿por qué entonces aparece rotado en posición ‘vertical’?

Antonia chasqueó los labios antes de contestar:

—Bien observado. Mirá, fue una decisión que tomaron los desarrolladores de OpenSCAD. Supongo que les habrá parecido más natural que la rotación ocurriera alrededor del canónico eje Z. Pero sinceramente no lo sé; el hecho es que es así —la última frase fue pronunciada por Antonia con tono de no admitir réplica. Cecilia, por su parte, tampoco encontró necesario discutirla.

»Si nuestra intención es obtener algo así como nuestro polígono rotado alrededor del eje Z, debemos crear otro que sea su estricta mitad en las Y positivas (o negativas) —dijo Antonia, ahora con tono negociador.

```

1  puntos2 = [[0,-20],
2               [0,20],
3               [20,20],
4               [10,0],
5               [20,-20]];
6
7  polygon(puntos2);

```

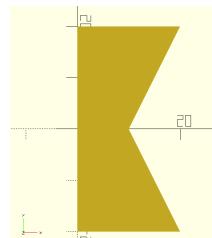


Figura 17.9: Antonia crea un nuevo polígono...

```

1 $fn=200;
2
3 rotate_extrude(angle=360)
4     polygon(puntos2);

```

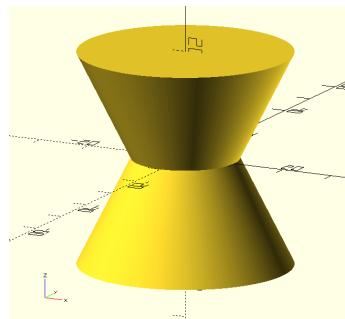


Figura 17.10: ...para que `rotate_extrude` cree con él un cuerpo que resulte una suerte de versión con simetría radial del polígono con el que comenzó el presente capítulo.

»Como sin duda ya habrás sospechado, el ángulo puede ser cualquiera —agregó Antonia, señalando la figura 17.11.

```

1 $fn=200;
2 rotate_extrude(angle=150)
3     translate([100,0,0])
4     polygon(puntos);
5
6 rotate_extrude(angle=250)
7     polygon(puntos2);

```

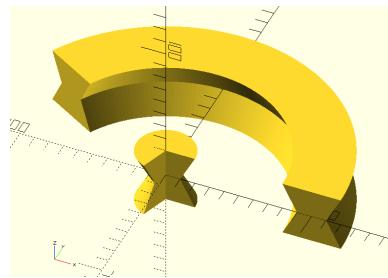


Figura 17.11: El ángulo usado por `rotate_extrude` no tiene por qué ser igual 360° .

17.4. DOS FIGURAS ESPECIALES

»Dos más y terminamos por hoy —aseguró Antonia—. Hay dos figuras especiales que seguramente querrás conocer: con

`circle(r=radio)` creás un círculo, y con `square([ancho, alto], center)` creás... bueno, creo que no hace falta que te lo diga.

```
1  rotate_extrude(angle=360)
2    translate([70,0,0])
3      circle(r=10);
```

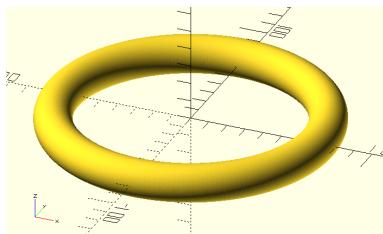


Figura 17.12: Antonia somete un `circle` a la transformación `rotate_extrude`.

```
1  rotate_extrude(angle=360)
2    translate([70,0,0])
3      square([10,30],
          center=true);
```

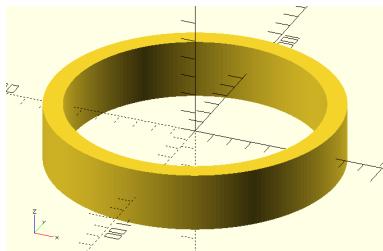


Figura 17.13: Antonia aplica la transformación `rotate_extrude` a un `square`.

Cecilia, a esta altura del capítulo, realmente ya no necesitaba que le dijera nada: sólo quería descansar.

El primer rayo de Sol – I

—VAMOS A VER ahora cómo escribir uno de los paralelepípedos con los que cortaremos el cuerpo de nuestro reloj; ¿te acordás? —preguntó Antonia al día siguiente. Cecilia, por su parte, no sólo lo recordaba perfectamente sino que estaba ansiosa por llevarlo a cabo.

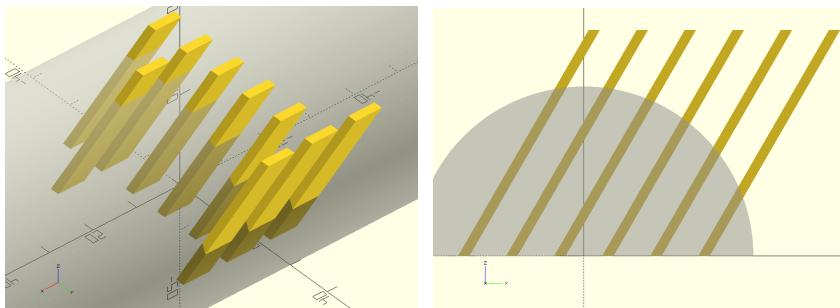


Figura 18.1: Antonia materializa los rayos de Sol como paralelepípedos que cortan el cuerpo del reloj para indicar así, bajo determinado ángulo, un dígito preciso: en este caso, el 1.

»Supongo que habrá otras maneras de resolverlo —Antonia se encogió de hombros—; pero a mí me resultó natural pensarlo

así: en primer lugar, llamé ‘pixeles’ a cada una de las caras que forman un dígito. ¡Qué se yo! Necesitaba un nombre, y me pareció tan bueno como cualquier otro, además de que me evocaba la idea de aquellos ‘puntos’ que conforman una imagen digital.

A Cecilia no le pareció que tal decisión mereciera objeción alguna, ni tampoco una defensa encendida.

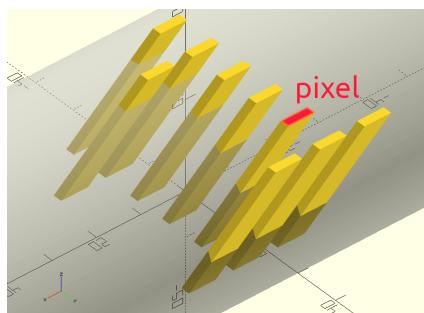


Figura 18.2: Antonia decide llamar “pixeles” a las caras superiores de los rayos de Sol que cortan el cuerpo del reloj.

—Cada pixel tendrá un ancho y un alto, que deberían poder ser elegidos por el usuario —afirmó Antonia dirigiendo la mirada de su amiga a la figura 18.3.

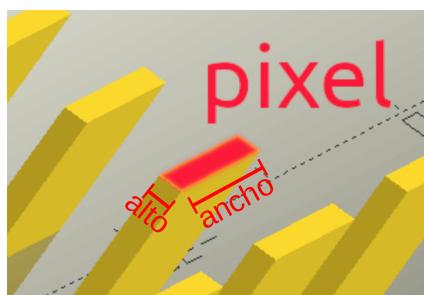


Figura 18.3: Los pixeles tendrán sus debidos ancho y alto.

»Ahora, me imaginé cada pixel extrudido a partir de un polígono con uno de los perfiles de la figura 18.4.

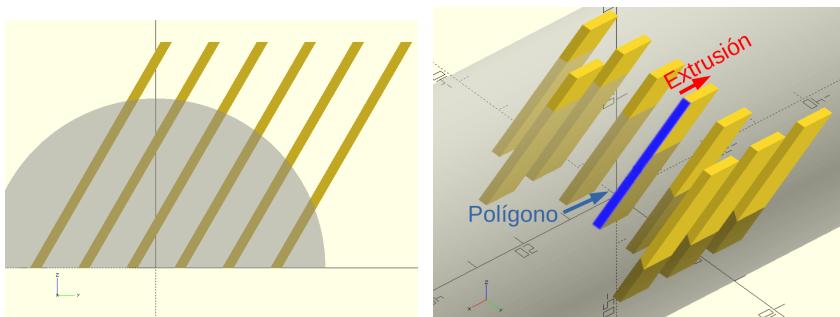


Figura 18.4: Antonia pretende erigir los rayos de Sol extruyendo polígonos, uno de los cuales destacó en azul.

»Y ahora, Cecilia —advirtió Antonia—, preparate porque se viene una andanada de pura geometría. Agarrate fuerte.

Cecilia quiso recordarle que estaba hablando con una doctora en ciencias; pero por un lado le pareció un poco pedante, y por otro no estaba tan segura de recordar cómo manejarse entre los vericuetos euclídeos en los que Antonia estaba a punto, sin duda, de aventurarse.

—Supongamos que ya calculamos la dirección en la que debe encontrarse el Sol al momento en que queremos que, debajo del reloj, aparezca el dígito deseado. Esa dirección estará determinada por un ángulo, que ya veremos cómo calcular gracias a nuestros fantásticos conocimientos astronómicos: llamémoslo α —dijo Antonia señalando la figura 18.5.

»Pues bien —Antonia parecía finalmente acercarse, tras algunos rodeos, a algún tipo de definición—; nuestro objetivo ahora es encontrar las coordenadas de los vértices del polígono que queremos luego extrudir: los destaque con circulitos azul pálido en la misma figura: ¿los ves?

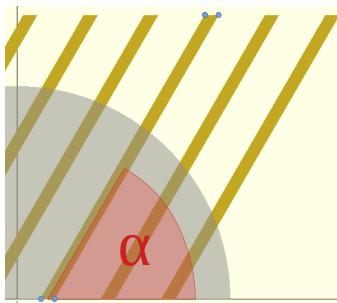


Figura 18.5: La inclinación de cada rayo estará determinada por la altura del Sol en cada momento, señalada con el ángulo α .

Cecilia los veía, pero no cómo calcularlos. «¿Trigonometría?» —pensó.

—Hay que usar trigonometría —soltó Antonia como si hubiera hecho una revelación—. No es tan difícil. Primero fijemos el punto origen del polígono; podría ser cualquiera, y a mí me resultó natural elegir el que señalé en la figura 18.6.

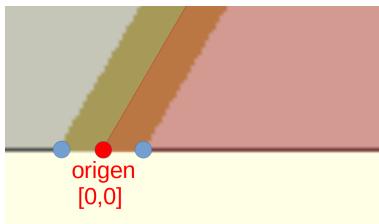


Figura 18.6: Antonia elige como origen de cada polígono el punto medio de su arista inferior.

»Sus coordenadas, por definición, son $[0,0]$: 0 en X y 0 en Y. Así, las coordenadas de los dos puntos de la base de nuestro polígono creo que son inmediatas —aseguró, señalando la figura 18.7.

Cecilia, al principio, se confundió un poco; pero recordó que



Figura 18.7: Antonia marca las coordenadas de los extremos inferiores del polígono buscado.

el ancho del polígono se correspondía con el alto del píxel final. Comprendió que, con respecto al origen elegido, ambos puntos tenían la misma coordenada Y (y, por lo tanto igual a 0), y diferían de él en una distancia igual a $\frac{\text{alto}}{2}$ en X, una con signo positivo y otra negativo.

—Muy bien —dijo Antonia, quizá no muy satisfecha con su propia explicación—; ahora veamos los dos vértices superiores. A mí, para fijar ideas, me resultó útil pensar en un punto auxiliar, ubicado justo en medio de dichos vértices, tal como se aprecia en la figura 18.8.

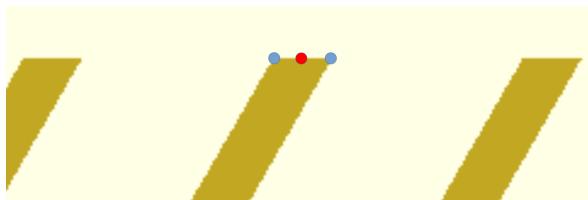


Figura 18.8: Antonia busca las coordenadas de los extremos superiores del polígono mediante un punto auxiliar, ubicado en medio de ambos.

»Podemos referirnos a la posición de ese punto auxiliar mediante dos valores: D y H. El primero será su distancia horizontal

con respecto al origen; en otras palabras, su coordenada X. H, análogamente, será su coordenada Y —explicó Antonia, señalando la figura 18.9.

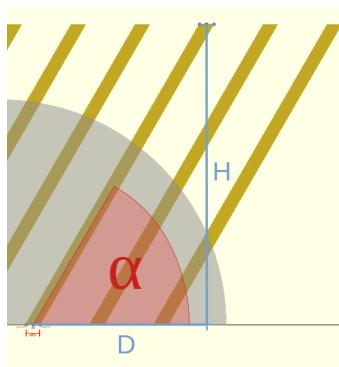


Figura 18.9: Antonia señala las coordenadas del nuevo punto auxiliar.

—¿Así que las coordenadas de tu nuevo punto auxiliar serán [D, H]? —preguntó Cecilia, no tanto para confirmar que entendía como para animar a su amiga a continuar.

—¡Exacto! —exclamó Antonia con el súbito entusiasmo que se apoderaba de ella cuando sucumbía a la ilusión de ser comprendida—. Ahora bien, habíamos quedado en que el valor de α era conocido por nosotras^{1,2}; de esa manera, la trigonometría nos promete que hay una firme relación entre α , D y H, gracias a la función tangente:

$$\tan \alpha = \frac{H}{D} \quad (18.1)$$

¹Espero que le dediquen un capítulo a esta delicada cuestión. (Nota del Editor)

²¡No te preocupes! ¡Lo haremos! ¡Qué ansioso..! (Nota de Antonia y Cecilia)

Cecilia recordaba claramente esa relación; supuso que OpenSCAD sabría cómo calcularla.

—OpenSCAD, por supuesto, cuenta entre sus funciones elementales la tangente —Antonia volvió a leer la mente de Cecilia, quien ya estaba comenzando a preocuparse por esa clarividencia tan oportuna. Por momentos le hacía pensar que su vida no era real, sino que formaba parte de un cuento, o tal vez un manual mal escrito.

—Hasta ahora, todo muy bonito, Antonia —interrumpió Cecilia, visiblemente incómoda—; comprendo que α lo podemos calcular astronómicamente, pero: ¿Qué onda D y H? Una única ecuación no nos permite calcular dos valores simultáneamente —preguntó, con tono casi acusador.

Antonia asintió suavemente con la cabeza, mientras fruncía los labios:

—Tenés razón; y acá tuve que tomar una decisión más. ¡Ay, Cecilia! —suspiró, reclinándose contra la silla—. ¡La programación, como la vida, exige constantemente de nosotras la toma de decisiones..! —Antonia se detuvo un momento, como buscando un recuerdo—. A ver si puedo mostrarte la manera en que lo pensé...

Tras escribir unos instantes, durante los cuales creó las imágenes de la figura 18.10, Antonia continuó:

—Aquí dispuse el mismo dígito, formado por cierto número de pixeles, con distintos ángulos α , correspondientes a diferentes posiciones del Sol —Antonia dirigió su mirada a Cecilia, y en sus ojos había una luz que parecía una súplica—. ¿Notás algún patrón que se repita en todos los casos..?

Cecilia sintió súbitamente sobre sus hombros una grave responsabilidad: no tanto por salvaguardar su reputación de “inteligente”, sino para que Antonia no sintiera que su explicación la conducía a una confusión irremontable. Como tantos otros alumnos antes y después que ella, sintió la piadosa obligación de

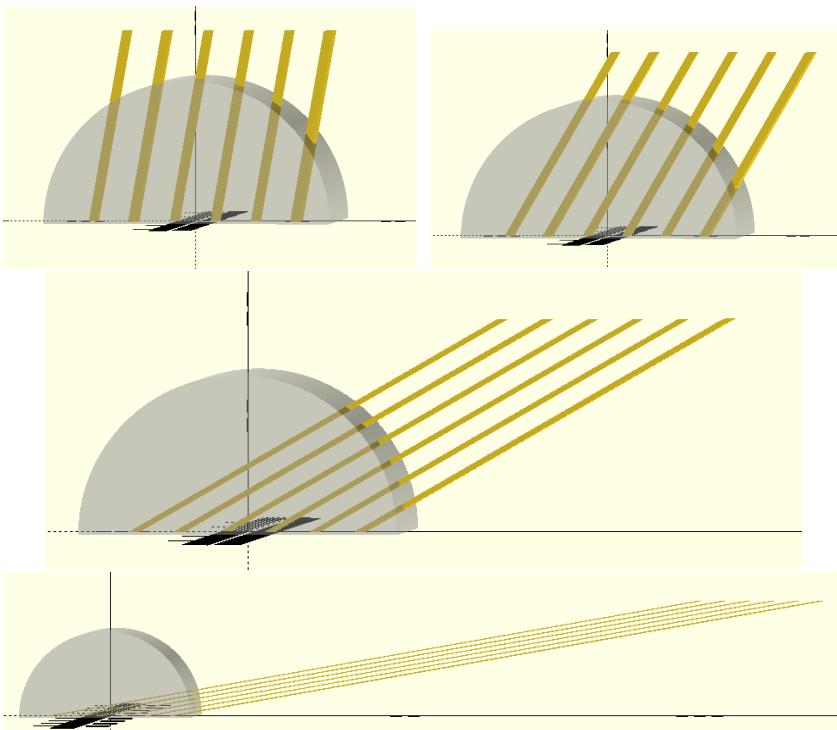


Figura 18.10: Antonia despliega para Cecilia distintos grupos de rayos con diferentes ángulos α , a fin de que descubra la relación que guarda el mismo con D y H en todos los casos.

justificar a su maestra. Con la mirada y la mente concentradas en el monitor, contempló los cuatro casos que Antonia le ofrecía.

«¿Qué tendrán en común estos dichosos ejemplos...?» —pensó, al borde del malhumor y de rendirse. Pero de pronto y sin saber de dónde, la epifanía ocurrió:

—¿Puede ser que todos los rayos tengan la misma altura? ¿Es decir, que comparten el mismo H, en todos los casos y con cualquier ángulo? —dijo.

El suspiro de alivio de Antonia debió haberse oído hasta en la oficina de Fumington:

—Sí, Cecilia... sí —Antonia estaba radiante—. El valor de H es convencional; es importante, eso sí, que sea mayor al radio del semicilindro, para que los rayos puedan cortarlo siempre. Es cierto —reconoció— que eso hará que los mismos resulten muy largos para ángulos muy bajos; pero, como ya conversamos alguna vez, rayos muy dilatados no ocupan más sitio en la memoria de la computadora que otros más cortos.

A Cecilia empezó a parecerle una solución muy astuta:

—Así que elegimos un H cualquiera y fijo, más grande que el radio del semicilindro, y con α y la ecuación 18.1 calculamos D —resumió, quizás para comprobar que había entendido.

Antonia aplaudió lentamente, mientras asentía con la cabeza. Cecilia no supo discernir para quién era la felicitación.

—Con lo visto, las coordenadas de los vértices superiores del polígono deberían resultar fáciles de determinar —confió Antonia.

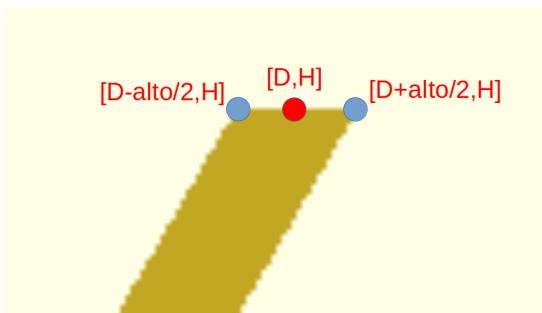


Figura 18.11: Antonia explicita las coordenadas de los vértices superiores.

»Así que nuestro rayo de Sol debería comenzar con un polígono cuyos vértices, referidos a un cierto origen de coordenadas, sean $[-\text{alto}/2, 0]$, $[D-\text{alto}/2, H]$, $[D+\text{alto}/2, H]$, $[\text{alto}/2, 0]$.

```

1 alto_pixel = 2;
2 ancho_pixel = 6;
3 radio_semicilindro = 30;
4 H = radio_semicilindro+10;
5
6 module rayo_de_sol(alfa){
7     D=H/tan(alfa);
8     vertices=[[-alto_pixel/2,0] ,
9                [D-alto_pixel/2,H] ,
10               [D+alto_pixel/2,H] ,
11               [alto_pixel/2,0]];
12 polygon(vertices);
13 }
14
15 rayo_de_sol(alfa=60);

```

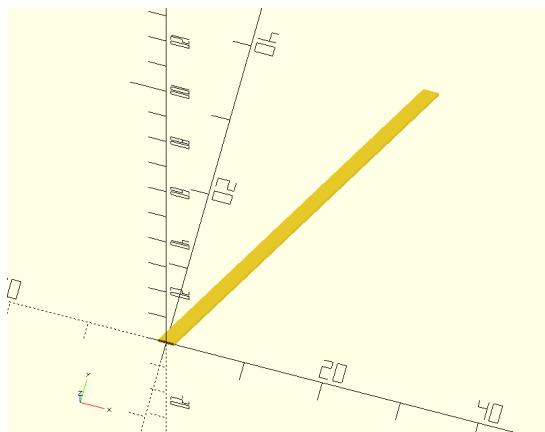


Figura 18.12: Antonia, finalmente, está en condiciones de escribir el código del polígono de un rayo de Sol.

Cecilia no pudo disimular su emoción: ¡Por fin algo de texto! Sintió que nada le confirmaría mejor que iban por un buen camino que el funcionamiento cabal de un programa que reflejara fielmente las ideas que elaboraban juntas.

—Por supuesto —aclaró Antonia—, el rayo no está terminado: por ahora es sólo un polígono recostado en el plano XY. Debemos convertirlo en un genuino objeto 3D mediante una extrusión lineal, y luego pararlo mediante una rotación.

—Sí, Antonia, entiendo; ¿pero no podemos dejarlo para el capítulo siguiente? Por éste creo que ya vimos mucho —suplicó Cecilia, quien estuvo a punto de utilizar la palabra “demasiado” en lugar del más diplomático “mucho”.

— 19 —

El primer rayo de Sol – II

—N^{os} QUEDÓ pendiente extrudir el polígono —Antonia empezo el capítulo yendo directamente al grano.

```
1 alto_pixel = 2;
2 ancho_pixel = 6;
3 radio_semicilindro = 30;
4 H = radio_semicilindro+10;
5
6 module rayo_de_sol(alfa){
7     D=H/tan(alfa);
8     vertices=[[ -alto_pixel/2,0] ,
9                [D-alto_pixel/2,H] ,
10               [D+alto_pixel/2,H] ,
11               [alto_pixel/2,0]];
12     linear_extrude(ancho_pixel)
13     polygon(vertices);
14 }
15
16 rayo_de_sol(alfa=60);
```

—Y ahora, ¡a rotarlo! —intervino Cecilia, riendo y tomando el teclado para sí.

```
1 alto_pixel = 2;
```

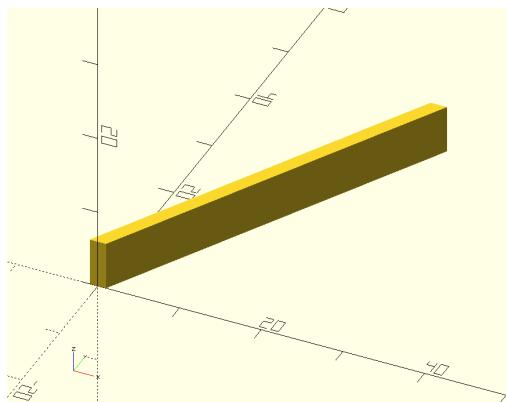


Figura 19.1: Rayo de Sol extrudido en espera de ser rotado.

```
2 ancho_pixel = 6;
3 radio_semicilindro = 30;
4 H = radio_semicilindro+10;
5
6 module rayo_de_sol(alfa){
7   D=H/tan(alfa);
8   vertices=[[ -alto_pixel/2,0],
9             [D-alto_pixel/2,H],
10            [D+alto_pixel/2,H],
11            [alto_pixel/2,0]];
12   rotate([90,0,0])
13   linear_extrude(ancho_pixel)
14   polygon(vertices);
15 }
16
17 rayo_de_sol(alfa=60);
```

Cecilia admiró feliz el resultado en la figura 19.2: estaban cada vez más cerca de su ansiado reloj de Sol digital. Sin embargo, un detalle congeló su sonrisa: advirtió que la base del rayo no estaba centrada en el origen, sino un poco corrida:

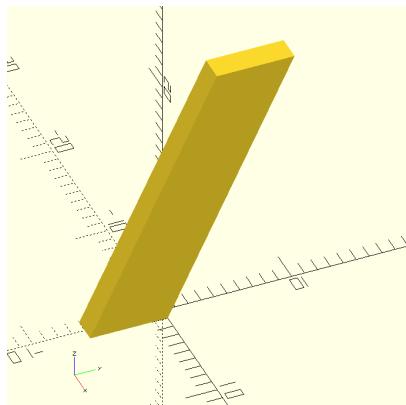


Figura 19.2: Primer rayo de Sol rotado por Cecilia.

—Antonia, el rayo nos quedó del lado de las Y negativas: ¿no será un problema? ¿Te parece que lo traslademos en `ancho_pixel/2` a lo largo de ese eje?

Antonia sonrió, enarcando las cejas con admiración:

—Bien por notar el problema, y mejor por encontrar una solución. En realidad, no sé si será un problema más adelante; en todo caso, siempre conviene que los módulos se comporten de la manera más simple posible. En este caso, todo parece indicar que lo más natural es que la base del rayo de Sol esté centrada —Antonia hizo una pausa, y su mirada parecía ver más allá del monitor—. Efectivamente, puede resolverse con una mera translación; basta intercalar la indicación `translate([0,ancho_pixel/2,0])` entre las líneas 11 y 12 de tu texto. Pero hay otra forma:

```
1 alto_pixel = 2;
2 ancho_pixel = 6;
3 radio_semicilindro = 30;
4 H = radio_semicilindro+10;
5
6 module rayo_de_sol(alfa){
```

```

7   D=H/tan(alfa);
8   vertices=[[-alto_pixel/2,0],
9             [D-alto_pixel/2,H],
10            [D+alto_pixel/2,H],
11            [alto_pixel/2,0]];
12  linear_extrude(ancho_pixel,center=true)
13    polygon(vertices);
14 }
15
16 rayo_de_sol(alfa=60);

```

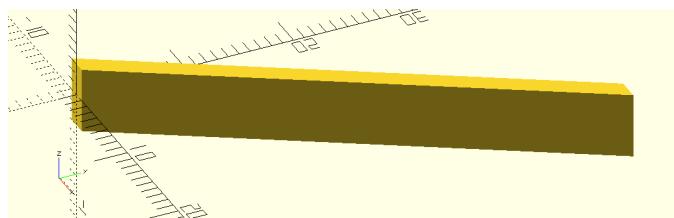


Figura 19.3: Antonia extruye el polígono del rayo de Sol con la opción `center=true`.

»Cuando agregás la prescripción `center=true` a la transformación `linear_extrude` (como hice en la línea 12), la extrusión se realiza centrada con respecto al plano XY. Ahora sí, si lo rotamos:

```

1 alto_pixel = 2;
2 ancho_pixel = 6;
3 radio_semicilindro = 30;
4 H = radio_semicilindro+10;
5
6 module rayo_de_sol(alfa){
7   D=H/tan(alfa);
8   vertices=[[-alto_pixel/2,0],
9             [D-alto_pixel/2,H],
10            [D+alto_pixel/2,H],
11            [alto_pixel/2,0]];
12   rotate([90,0,0])

```

```

13     linear_extrude(ancho_pixel,center=true)
14         polygon(vertices);
15 }
16
17 rayo_de_sol(alfa=60);

```

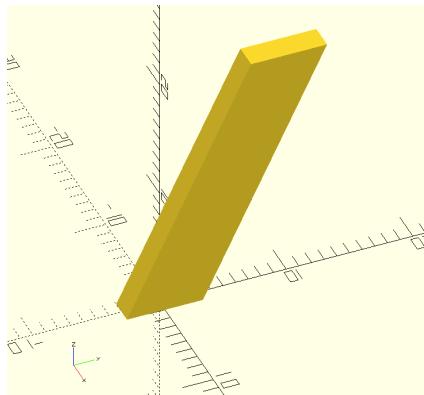


Figura 19.4: Rayo de Sol correctamente centrado.

Cecilia aplaudió con indisimulada alegría la figura 19.4. Antonia seguía concentrada con la mirada en dirección al monitor:

—Te confieso que no estoy segura de cuál de las dos soluciones sea la mejor. Deberíamos tener en cuenta que nuestro reloj estará atravesado por multitud de rayos, todos los cuales deberán ser calculados por la computadora: siempre es bueno, en esos ‘cuellos de botella’ computacionales, elegir el algoritmo más eficiente —Antonia suspiró—. No conozco cómo funciona internamente el motor de OpenSCAD; supongo que lo mejor que podemos hacer es probar ambas soluciones, y medir cuánto tarda cada una. Es un buen momento para colocar un comentario al respecto:

```

1 module rayo_de_sol(alfa){
2     D=H/tan(alfa);
3     vertices=[[-alto_pixel/2,0],

```

```
4           [D-alto_pixel/2,H],  
5           [D+alto_pixel/2,H],  
6           [alto_pixel/2,0]];  
7 // TODO: medir la duracion de esta solucion  
8 //         y la de esta otra:  
9 // translate([0,ancho_pixel/2,0])  
10 //         Ojo: borrar 'center=true' abajo  
11 rotate([90,0,0])  
12 linear_extrude(ancho_pixel,center=true)  
13 polygon(vertices);  
14 }
```

Cecilia estaba demasiado feliz como para detenerse en esos detalles; además, otra idea súbitamente se impuso a su atención.

19.1. ¿UN RAYO DE SOL ALTERNATIVO?

—¡Antonia! ¡Pará! —exclamó Cecilia, con tono vibrante—. Dejame probar... A ver... ¿No se puede hacer un rayo de Sol así..? —y tomó el teclado con decisión, mientras daba forma a su incipiente idea escribiéndola.

```
1 module otro_rayo_de_sol(alfa) {  
2     rotate([-90-alfa],0,0)]  
3     translate([-ancho_pixel/2,-alto_pixel/2,0])  
4     cube([ancho_pixel,alto_pixel,H]);  
5 }  
6  
7 otro_rayo_de_sol(alfa=60);
```

Cecilia se mostraba decididamente radiante mientras contemplaba la figura 19.5. «¿Para qué usar un polígono» —pensó— «si podemos usar un elemental cubo?». Esperó una felicitación de Antonia, o al menos un comentario, pero en vano: cuando volteó para mirarla, encontró en ella sólo un gesto de displicente reserva.

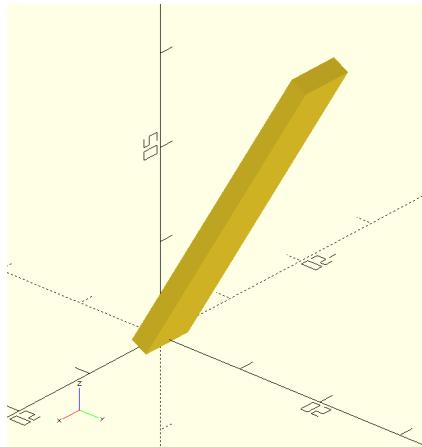


Figura 19.5: Un rayo de Sol alternativo, propuesto por Cecilia con entusiasmo.

—Yo también caí en esa trampa al principio —confesó Antonia—. La idea de un cubo se nos presenta eficaz, incluso elegante; pero un detalle nefasto me demostró toda su perversidad.^{1,2,3} Dejame que te muestre... —Antonia se puso a escribir, frente la atenta mirada de su amiga.

»En la figura 19.6 te agregué un piso para que vieras la marca que tu rayo de Sol, una vez atravesado el reloj, dejaría en el mismo. ¿No notás nada raro? —preguntó Antonia.

Cecilia respondió encogiéndose de hombros y enarcando las cejas.

—Sí, entiendo —dijo Antonia—; no es tan evidente aún. A ver si lo miramos bien de costado —agregó, produciendo la figura 19.7.

—Hmmm... tampoco —confesó Cecilia.

¹Eh..! ¿Será para tanto? (Nota del Editor)

²No: me agarró un ataque de retórica, nomás. (Nota de Antonia)

³Ah, listo; todo bien. (Nota del Editor)

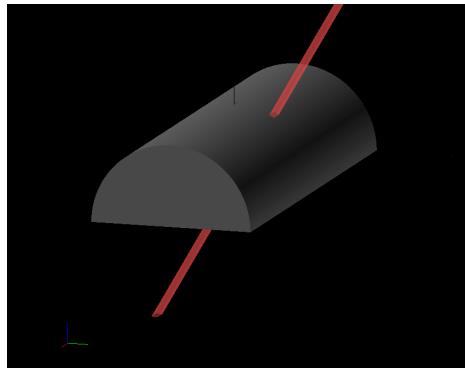


Figura 19.6: Primer ejemplo con el que Antonia trata de disuadir a Cecilia de emplear su rayo de Sol alternativo.

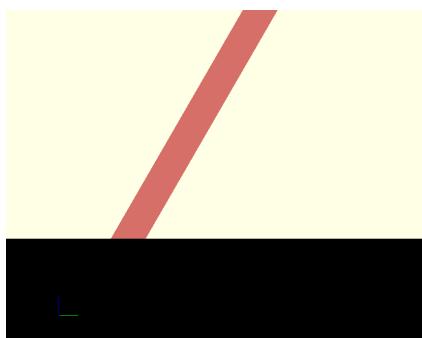


Figura 19.7: Antonia sigue tratando de disuadir a Cecilia de emplear su rayo de Sol alternativo.

—¿Y con un ángulo más bajo..? —intentó Antonia ahora con la figura 19.8.

A Cecilia le pareció empezar a comprender.

—¿Y con 10° ..? —Antonia suplicó señalando la figura 19.9.

—¡Claro! —Cecilia se golpeó suavemente la frente con una mano—. ¡La mancha de luz en el piso comienza a alargarse! Lo

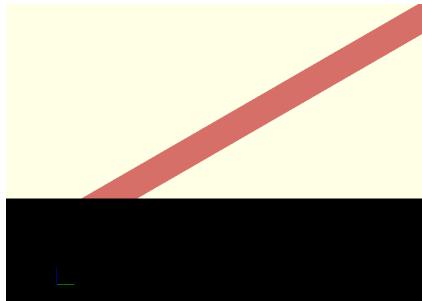


Figura 19.8: Antonia no ceja en su intento de disuadir a Cecilia de emplear su rayo de Sol alternativo.

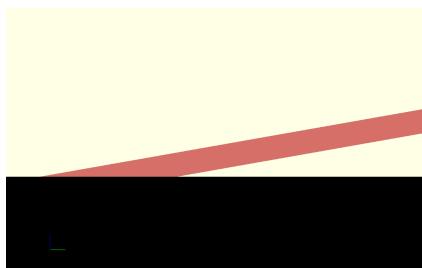


Figura 19.9: Último intento de Antonia para disuadir a Cecilia de emplear su rayo de Sol alternativo.

que debe medir `alto_pixel` es la intersección del rayo con la horizontal, no su sección transversal; por eso dibujaste el polígono como lo hiciste.

—Por eso dibujé el polígono así *al final* —Antonia subrayó estas últimas palabras—. Al principio lo hice como vos. Ésa es una de las razones por las que me gusta escribir: aclara mis ideas.

Cecilia y Antonia se miraron, sonriendo.

Un dígito hecho de Sol – I

—CADA UNO DE los dígitos que indicarán la hora estarán formados por rayos de Sol —arrancó Antonia, ofreciendo a Cecilia la figura 20.1.

—¡Qué buenos diseños! ¿Los hiciste vos? —preguntó Cecilia.

—No —reconoció Antonia sin dificultad—. La disposición visual la copié inescrupulosamente del modelo que me inspiró en general: <https://www.thingiverse.com/thing:1068443>.

—¡Ah! Sos pilla... —Cecilia sonrió con malicia.

—Fue sólo el aspecto visual —Antonia se refugió en una actitud defensiva—. La lógica y el texto los reconstruí yo sola.

Cecilia miró unos instantes a su amiga detenidamente; no sabía si hacerle una pregunta. Finalmente la soltó:

—Antonia... ¿Por qué rehiciste un modelo que ya fue resuelto por otra persona? ¿No te bastaba bajarlo e imprimirlo?

Antonia devolvió a Cecilia la mirada:

—¿Por qué estás haciendo este curso? —replicó—. ¿Por qué llegaste a la página 163 de este manualcito?

Las dos amigas se miraron en silencio. Cecilia comprendió que pocas cosas se comparaban con resolver, por una misma, un problema difícil y con un resultado final de aspecto hermoso. Y el reloj de Sol digital combinaba ambas cualidades a la perfección.

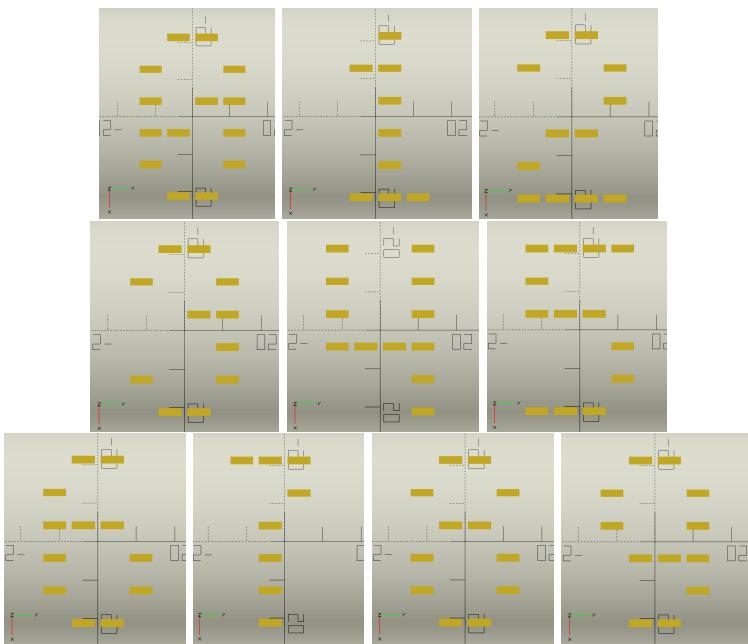


Figura 20.1: Dígitos hechos de rayos de Sol.

—Como habrás notado —retomó Antonia—, todos los dígitos se basan en un mismo patrón de seis filas y cuatro columnas de pixeles. La diferencia entre los dígitos reside en los pixeles que están ‘encendidos’ o ‘apagados’; aunque sería mejor decir ‘abiertos’ o ‘cerrados’, dado que cada uno representa un agujero en el semicilindro del reloj.

—El ‘1’ parece contar con tres columnas, solamente —observó Cecilia.

—La cuarta columna está, sólo que formada por pixeles ‘cerrados’ —aclaró Antonia. A Cecilia le pareció un poco artificial. «¿Y por qué no pensar, entonces, que está formado por 17 filas y 48 columnas, la mayoría de ellas ‘cerradas’?» —pensó, pero la dejó

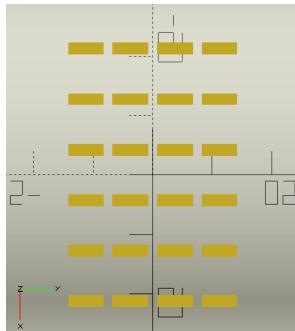


Figura 20.2: Matriz de píxeles que formarán cada dígito.

continuar.

—Para aclarar ideas, vamos a ver todos los rayos ‘encendidos’ —dijo Antonia señalando la figura 20.2, mientras prosiguía con el lento tono que presagiaba una de sus largas explicaciones—. Cada dígito, entonces, se formará dejando abiertos o cerrados los píxeles adecuados. Pero antes de ver cómo elegir cada uno, creo que es mejor que resolvamos el problema de ubicar cada rayo en una disposición ordenada y prolífica de filas y columnas.

Cecilia puso cara de inteligente:

—¿Usamos la transformación [translate](#)?

—Sí, más vale; pero la pregunta es *¿cómo?* —Antonia pareció no haber captado la ironía—. A mí me sirve empezar haciendo dibujitos encima —agregó, realizando la figura 20.3.

»Para poder distribuir correctamente los píxeles, debemos ponernos de acuerdo no sólo en el tamaño de cada pixel (expresado en nuestros conocidos `ancho_pixel` y `alto_pixel`) sino también en la separación entre píxeles: se me ocurrió llamarlos `delta_ancho` y `delta_alto`... básicamente porque sí.

Cecilia no tuvo nada que objetar.

—Ahora tenemos que poder distinguir a cada pixel de los demás. Ponerles un ‘nombre’, digamos. Se me ocurrió que hacerlo

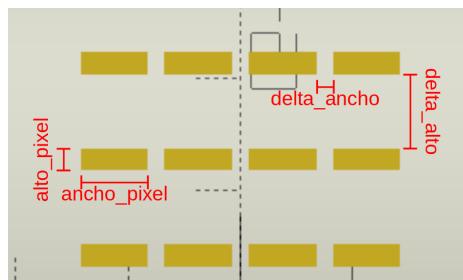


Figura 20.3: Antonia anota la matriz con el fin de ubicar cada pixel.

en referencia a la fila y columna que ocupan en el arreglo no puede ser la peor forma —dijo Antonia.

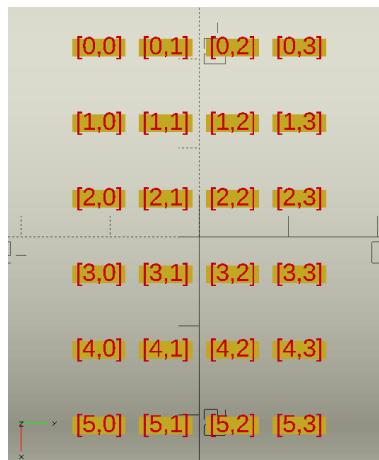


Figura 20.4: Antonia nombra cada pixel con las coordenadas enteras de su matriz.

»Así, el primer pixel será el de la fila 0 y columna 0, etc. Como comentamos en el capítulo 10, es usual referirse a las posiciones dentro de un arreglo rectangular empezando por 0 —recordó

Antonia, y prosiguió—: Ahora, lo más difícil: ubicar la posición en X e Y de cada pixel conociendo su fila y columna, tomando en cuenta los valores de `ancho_pixel`, `alto_pixel`, `delta_ancho` y `delta_alto`.

»Para ir paso a paso, me parece mejor ubicar como origen de coordenadas el centro del pixel $[0, 0]$. Sí, ya sé —se adelantó—; nosotras queremos que sea el centro del arreglo el que esté en el origen de coordenadas; pero creo que es más fácil comenzar a pensarlo así.

Cecilia no se sentía con muchas ganas de contradecir a Antonia, básicamente porque aún no sabía exactamente adónde se dirigía en ese camino que ya le parecía demasiado largo.

—Muy bien, Cecilia; ya hablé mucho —Antonia miró a su amiga con los ojos apenas entornados y con un brillo que, una vez más, parecía una súplica—. Ahora tendríamos que ser capaces de calcular el valor de X e Y para cualquier pixel: digamos, por ejemplo, el $[3, 2]$.

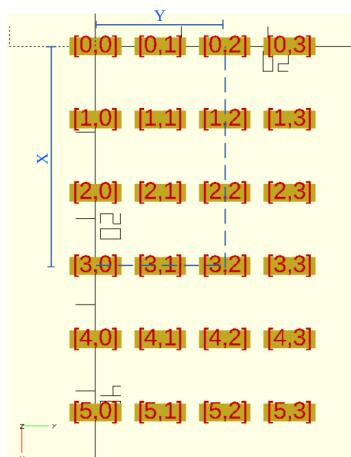


Figura 20.5: Antonia desafía a Cecilia a encontrar las coordenadas X e Y de cada pixel; en particular, del $[3, 2]$.

Antonia hizo silencio, y Cecilia supo que aquel plural era, como otras veces, bien singular.

—¿Me dejás hacer unos dibujitos? —preguntó Cecilia mientras tomaba para sí el teclado.

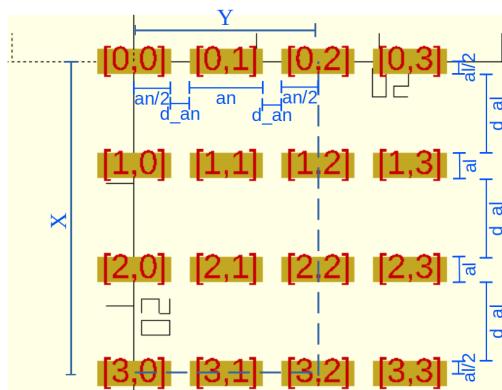


Figura 20.6: Cecilia procura expresar las coordenadas X e Y de cada pixel, comenzando por el [3, 2].

»¡Uf! Me costó... —reconoció mientras se reclinaba pesadamente contra el respaldo de la silla y contemplaba la flamante figura 20.6—. Como no me entraba en el dibujo, tuve que usar abreviaturas: an es ancho_pixel, d_an es delta_ancho... en fin, creo que lo demás se entiende.

Antonia asintió con entusiasmo.

—Por lo que veo —Cecilia continuó—, la coordenada Y del pixel [3, 2] es igual a $\frac{an}{2} + d_an + an + d_an + \frac{an}{2} = 2 \times an + 2 \times d_an = 2 \times (an + d_an)$. Por otra parte, la coordenada X parece ser igual a... —Cecilia masculló apenas los pasos intermedios—: $3 \times (al + d_al)$.

Antonia, con una amplia sonrisa de satisfacción, tomó la palabra y el teclado:

—Pasemos en limpio, entonces:

$$\text{Para el pixel } [3, 2] : \begin{cases} X = 3 \times (al + d_al) \\ Y = 2 \times (an + d_an) \end{cases}$$

Cecilia contemplaba las ecuaciones y el grafico. No pasó mucho tiempo hasta que encontró el patrón subyacente:

—¡Claro! —exclamó exultante—. La coordenada X del pixel $[i, j]$ es i veces $(al + d_al)$, mientras que la coordenada Y, analógicamente, es j veces $(an + d_an)$.^{1,2}

Antonia confirmó la conclusión de Cecilia con indisimulada alegría:

—Vamos a dejarlo bien patente —propuso.

$$\text{Para el pixel } [i, j] : \begin{cases} X = i \times (al + d_al) \\ Y = j \times (an + d_an) \end{cases} \quad (20.1)$$

»Creo que es hora de darle a tantas vueltas y rodeos la forma de un texto para OpenSCAD, ¿no te parece? —invitó Antonia, cediéndole el teclado con gesto ceremonioso.

Cecilia lo tomó con decisión, aún no del todo segura de lo que iba a escribir, pero con entusiasmo y la confianza de que las teclas  y  le serían fieles. Tras mucho escribir, borrar y consultar con las ecuaciones, obtuvo lo siguiente:

```

1 alto_pixel  = 2;
2 ancho_pixel = 6;
3 delta_alto  = 6.5;
4 delta_ancho = 1.5;
5
6 module digito(alfa){
7   for(i=[0:5],j=[0:3]){
8     x=i*(alto_pixel+delta_alto);

```

¹Sería bueno demostrar fehacientemente un aserto tan general (Nota del Editor).

²¿No tenés nada mejor que hacer? (Nota de Cecilia, Antonia y Luis).

```

9     y=j*(ancho_pixel+delta_ancho);
10    translate([x,y,0])
11    rayo_de_sol(alfa);
12 }
13 }
14
15 digito(alfa=90);

```

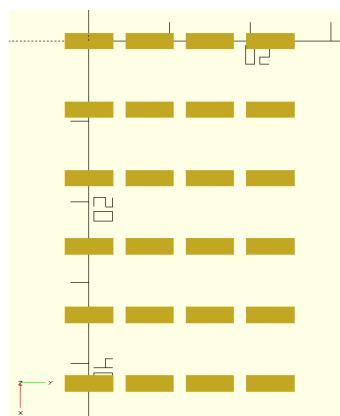


Figura 20.7: Cecilia logra producir una matriz completa de pixeles encendidos.

Cecilia se sentía tan feliz contemplando el resultado plasmado en la figura 20.7 que pensaba que podría releer su texto hasta que el Sol se apagara.

Dentro del módulo creaba un doble bucle: las variables *i* y *j* recorrían los valores de 0 a 5 y de 0 a 3, respectivamente, a fin de abarcar todas las filas y columnas.

Luego, en las líneas 8 y 9 calculaba las coordenadas *X* e *Y* de cada pixel, aplicando las ecuaciones 20.1.

Por último, en las líneas 10 y 11 creaba para cada pixel un rayo de Sol trasladado a las coordenadas recién calculadas.

—Parece mentira que además funcione, ¿no? —Antonia inter-

rrumpió los pensamientos de Cecilia—. A mí me pasa lo mismo: una cosa es *saber* que un algoritmo debe funcionar, y otra muy distinta es *comprobar* que efectivamente funciona. Es... hermoso.

Cecilia estaba de acuerdo de todo corazón. Pero Antonia se reservaba una objeción:

—Fíjate la posición del arreglo completo: te quedó centrado en su pixel [0,0], no en su centro.

Cecilia se mordió el labio inferior; la ansiada versión final del texto siempre parecía alejarse un poco más.

—Es cierto —admitió—; pero parece sencillo de resolver: no hay más que trasladar todo una cierta distancia en X e Y... a ver...

—y Cecilia, en breves instantes, pudo ofrecer su nueva solución:

```
1 module digito(alfa){  
2     translate([-2.5*(alto_pixel+delta_alto),  
3                 -1.5*(ancho_pixel+delta_ancho),  
4                 0])  
5     for(i=[0:5],j=[0:3]){  
6         x=i*(alto_pixel+delta_alto);  
7         y=j*(ancho_pixel+delta_ancho);  
8         translate([x,y,0])  
9         rayo_de_sol(alfa);  
10    }  
11 }  
12  
13 digito(alfa=90);
```

Antonia aprobó la modificación con una sonrisa y un aplauso apagado: la traslación del arreglo ocurría gracias al `translate` de las líneas 2 a 4, que afectaba a todo lo creado por el `for` inmediato siguiente.

Pero ahora fue Cecilia quien parecía mirar su propio texto con ojos críticos: las líneas 2 y 3 le parecieron demasiado similares a las 6 y 7. «¿No estoy trasladando dos veces, y con una lógica similar, los mismos rayos de Sol?» —pensó—. «¿Por qué no hacerlo de una sola vez...?». Y se lanzó a plasmar su idea:

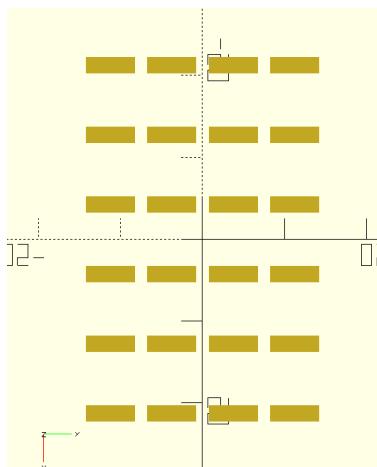


Figura 20.8: Cecilia centra debidamente la matriz en el origen de coordenadas.

```

1 module digito(alfa){
2     for(i=[0:5],j=[0:3]){
3         x=(i-2.5)*(alto_pixel+delta_alto);
4         y=(j-1.5)*(ancho_pixel+delta_ancho);
5         translate([x,y,0])
6         rayo_de_sol(alfa);
7     }
8 }
9
10 digito(alfa=90);

```

—Brillante, Cecilia, brillante... —Antonia quizá exageraba, pero a Cecilia no le importó: estaba muy orgullosa de su texto—. Esencialmente, sacaste factor común a las ecuaciones de las dos traslaciones. Después de todo, $-2,5 \times (\text{alto_pixel} + \text{delta_alto}) + i \times (\text{alto_pixel} + \text{delta_alto}) = (i - 2,5) \times (\text{alto_pixel} + \text{delta_alto})$, y lo mismo vale para Y.

Cecilia creyó que ya podían dar por finalizado el capítulo, pero

Antonia parecía querer alargarlo un poco más:

—Nos queda un detalle, que es mejor que resolvamos ahora. Como recordarás del capítulo 14, cuando restás un objeto de otro es conveniente que no compartan una cara en común, a fin de evitar una ambigüedad con respecto a la pertenencia o no de esa cara al objeto final. Y resulta que tus rayos de Sol se apoyan sobre el plano XY, al igual que el semicilindro que deberán horadar:

```
1 difference(){  
2     rotate([-90,0,0])  
3     semicilindro(150,30,center=true);  
4     digito(alfa=90);  
5 }
```

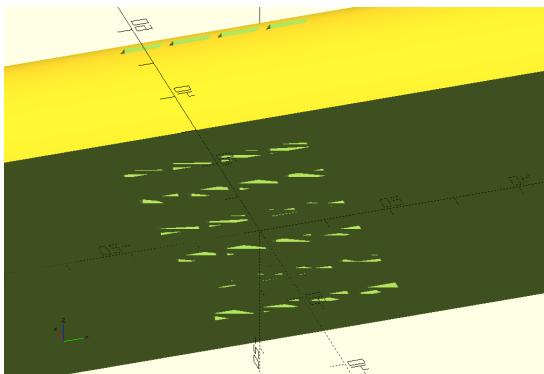


Figura 20.9: Antonia muestra un *bug* sutil del recientemente conquistado rayo de Sol.

—Noooo.... —gimió Cecilia, cubriendo sus ojos con una mano. La programación comenzó a dejar de parecerle divertida.

—Pero Cecilia... ¡no te vas a acorbadar por tan poco! —Antonia la animó sin mucho tacto—. La hacemos fácil: bajamos cada rayo un toque. Muy poquito; apenas como para que la matemática de las diferencias de OpenSCAD no se rompa:

```

1  module digito(alfa){
2      for(i=[0:5] ,j=[0:3]){
3          x=(i-2.5)*(alto_pixel+delta_alto);
4          y=(j-1.5)*(ancho_pixel+delta_ancho);
5          translate([x,y,-0.01])
6              rayo_de_sol(alfa);
7      }
8  }
9
10 difference(){
11     rotate([-90,0,0])
12     semicilindro(150,30,center=true);
13     digito(alfa=90);
14 }
```

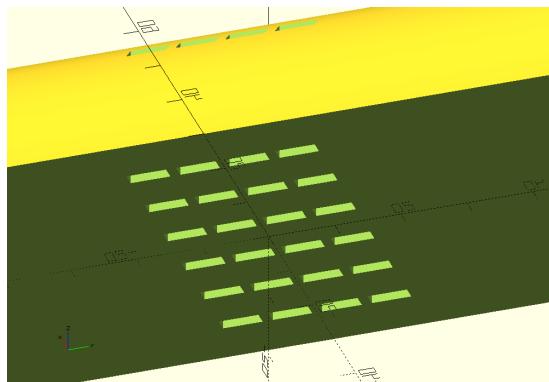


Figura 20.10: Antonia resuelve el *bug* con un expediente sencillo pero eficaz.

—¿Ves? —Antonia procuró parecer despreocupada, aunque tal vez sobreactuando un poco—. No se trata más que de trasladar cada rayo, en la línea 5, -0,01mm en el eje Z...

Cecilia podía apreciar en la figura 20.10 que la solución de Antonia funcionaba, pero no estaba muy convencida:

—¿Y por qué 0,01?

—Pues... con que se trate de un valor negativo alcanza, y 0,01 es mucho más bajo que la resolución de nuestra impresora 3D: en lo que respecta al objeto que produciremos, la diferencia con 0 es meramente formal —Antonia trató de que su tono confiado pareciese un argumento. Cecilia juzgó que se trataba de una solución chapucera, pero no podía negar que funcionaba. Y la verdad es que ya estaba cansada.

Pero antes de terminar quiso comprobar que nada se rompía usando otros ángulos, como 60° y 150° .

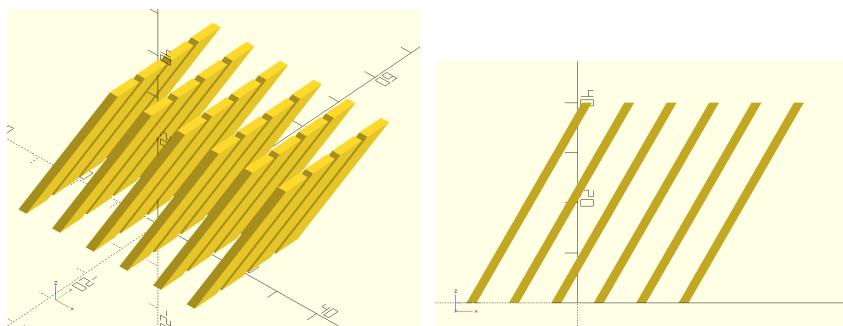


Figura 20.11: Cecilia comprueba que el rayo de Sol funciona con 60° : `digito(alfa=60)` ;...

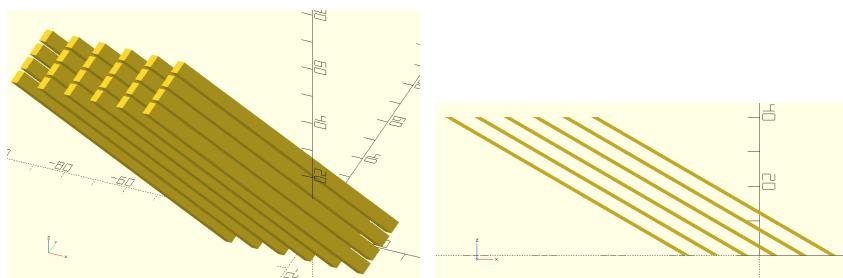


Figura 20.12: ...y con 150° : `digito(alfa=150)` ;

Parecía que no.

Un poco de lógica

—A HORA QUE YA sabemos prender todos los píxeles de un arreglo rectangular, debemos ser capaces de elegir cuáles apagar y cuáles dejar encendidos —Antonia comenzó sin rodeos el capítulo.

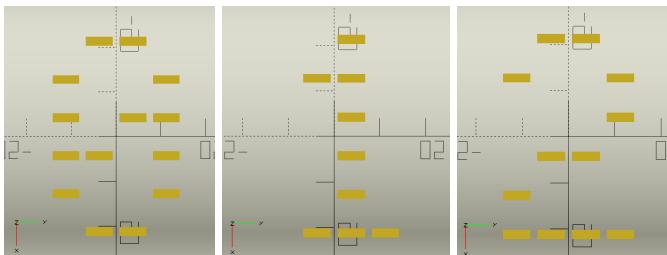
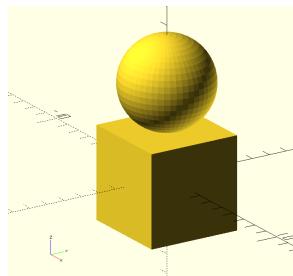


Figura 21.1: Algunos dígitos solares.

21.1. CONDICIONES

»Para crear objetos de acuerdo a la veracidad o no de una cierta condición se emplea la sentencia `if` —dijo Antonia mientras escribía el ejemplo de la figura 21.2.

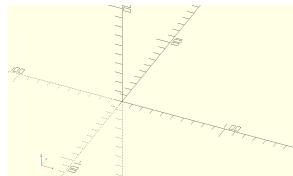
```
1 $fn=50;
2
3 if (1+1==2) {
4     cube([5,5,5],
5         center=true);
6     translate([0,0,5])
7         sphere(r=2.5);
```

Figura 21.2: Primer ejemplo del uso de `if`.

»`if` debe ir seguido por una condición a verificar escrita entre paréntesis. En el caso del ejemplo, se trata de la indiscutible aseveración de que $1 + 1 = 2$.¹

—¿Por qué en la línea 3 usaste dos veces el signo '='? —preguntó Cecilia.

```
1 $fn=50;
2
3 if (2+2==5) {
4     cube([5,5,5],
5         center=true);
6     translate([0,0,5])
7         sphere(r=2.5);
```

Figura 21.3: Resultado de un `if` falaz.

—Porque un solo signo '=' se usa exclusivamente para las asignaciones de variables —justificó Antonia, y prosiguió—: Si la condición es verídica, OpenSCAD pasa a concretar los objetos

¹Alfred North Whitehead y Bertrand Russell ofrecen una demostración de tan particular aserto en la página 86 del segundo volumen de sus *Principia Mathematica*. El primer volumen consta de unas 700 páginas. (Nota del Editor)

escritos entre llaves. Si no, los ignora olímpicamente, como podés comprobar en la figura 21.3.

21.2. ELSE

»También puede resultar útil, a veces, indicar diversos objetos a realizar tanto si una condición es cierta como si es falsa:

```

1 $fn=200;
2
3 if (1<0) {
4   cube ([5,5,5]);
5 } else {
6   sphere (r=2.5);
7 }
```

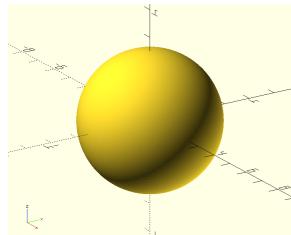


Figura 21.4: Uso de la construcción `if/else`.

»En el ejemplo de la figura 21.4, si 1 fuera menor a 0 hubiéramos visto aparecer un cubo; pero como la matemática nos promete que no es así, fue creada una esfera —claró Antonia—. Es posible, incluso, escoger entre más de dos posibilidades con el empleo del giro lingüístico `else if`.

Cecilia no necesitó en este caso de la explicación de Antonia al enfrentarse a la figura 21.5: pudo ver que si $1 + 1$ hubiera sido menor que 2, habría sido creado un cubo. Como no fue así, OpenSCAD pasó a comprobar si $1 + 1$ era mayor que 2. En caso afirmativo, hubieran visto una esfera; pero como no era el caso, se verificó luego si $1 + 1$ era igual a 2. Ante la indiscutible veracidad del hecho, OpenSCAD no tuvo más remedio que crear un cilindro.

—¿Qué pasa si varias condiciones son ciertas? —preguntó Cecilia.

—Se crean los objetos sujetos a la primera que lo sea; los demás

```
1 $fn=200;  
2  
3 if (1+1<2) {  
4   cube([5,5,5]);  
5 } else if (1+1>2) {  
6   sphere(r=2.5);  
7 } else if (1+1==2) {  
8   cylinder(h=5,r=2.5);  
9 }
```

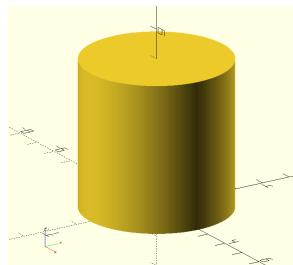


Figura 21.5: Antonia propone un ejemplo de la construcción `else if`.

se ignoran —respondió Antonia. Sin embargo, tras un momento, añadió: —Bah, así debería ser, supongo; vamos a comprobarlo por las dudas.

```
1 $fn=200;  
2  
3 if (1+1==2) {  
4   cube([5,5,5]);  
5 } else if (2+2==4) {  
6   sphere(r=5);  
7 } else if (1*1==1) {  
8   cylinder(h=5,r=5);  
9 }
```

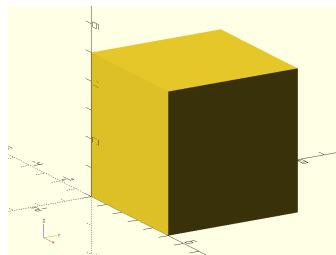


Figura 21.6: Antonia comprueba una conjetura acerca del funcionamiento de varios `else if`.

»Y así era, nomás —concluyó con una sonrisa ante la figura 21.6.

21.3. CONECTORES LÓGICOS

—Hay más comparadores, por supuesto, como ' $>=$ ' (mayor o igual), ' $<=$ ' (menor o igual) y ' $!=$ ' (distinto) —prosiguió Antonia—. Por otra parte, te van a resultar extremadamente útiles los conectores lógicos 'Y' (`&&`) y 'O' (`||`). El primero obliga a verificar que *todas* las condiciones entre paréntesis sean ciertas: si una sola es falsa, los objetos se omiten:

```
1 if (1+1==2 && 2+2==4 && 1<0) {
2   cube([5,5,5]);
3 }
```

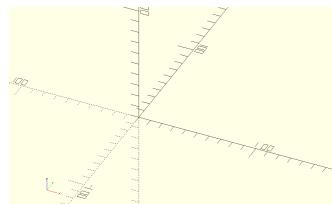


Figura 21.7: El conector lógico `&&` sólo da por cierto un conjunto de condiciones si todas lo son.

»El cubo hubiera aparecido en la figura 21.7 si $1 + 1 = 2$ y $2 + 2 = 4$ y $1 < 0$; pero como claramente la última condición es falsa, no apareció nada —comentó Antonia—. Por su parte, el conector `||` verifica que al menos una de las condiciones sea cierta; en ese caso, realiza los objetos entre llaves:

```
1 if (1+1==3 || 2+2==5 || 1>0) {
2   cube([5,5,5]);
3 }
```

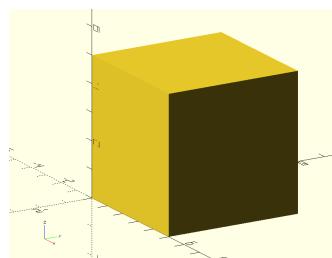


Figura 21.8: El conector lógico `||` da por bueno un conjunto de condiciones con tal que al menos una sea veraz.

»1 + 1 no es igual a 3, ni 2 + 2 será nunca 5; pero como 1 es mayor que 0, ahí tenemos nuestro cubo —redondeó Antonia señalando la figura 21.8—. Este capítulo fue cortito, así que aprovecho para comentarte que si el objeto a crear es uno solo, podés omitir las llaves, tal como se aprecia en la figura 21.9.

```
1 if (1+1==2)
2   cube([5,5,5]);
```

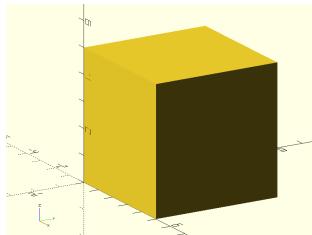


Figura 21.9: Si el `if` rige la creación de un solo objeto, las llaves pueden omitirse.

Cecilia ya casi no escuchaba a Antonia, porque su mente empezó a soñar con aprovechar lo visto en el presente capítulo para prender o apagar los pixeles de cada dígito solar.

Un dígito hecho de Sol – II

CECILIA VOLVIÓ A considerar los dígitos hechos de pixeles abiertos y cerrados en una disposición matricial. Sabía que de alguna manera debía decidir si cada pixel tenía que estar abierto o cerrado. Pero no acertaba a saber cómo.

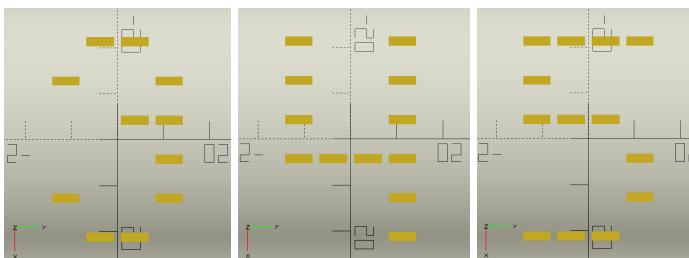


Figura 22.1: Algunos dígitos solares.

—El asunto no es sencillo —concedió Antonia—; pero por eso mismo es más interesante.

Tras una pausa, en la que pareció buscar la mejor manera de ayudar a Cecilia sin caer en un *spoiler*, continuó:

—Deberíamos ser capaces de recorrer cada fila y columna de la matriz completa, pero en cada paso decidir (mediante un `if`) si el pixel en cuestión debe ser creado o no. Te recuerdo el módulo

que creaba toda la matriz; lo escribimos en el capítulo 20:

```
1 alto_pixel = 2;
2 ancho_pixel = 6;
3 delta_alto = 6.5;
4 delta_ancho = 1.5;
5
6 module digito(alfa){
7     for(i=[0:5],j=[0:3]){
8         x=(i-2.5)*(alto_pixel+delta_alto);
9         y=(j-1.5)*(ancho_pixel+delta_ancho);
10        translate([x,y,-0.01])
11        rayo_de_sol(alfa);
12    }
13 }
14
15 digito(alfa=90);
```

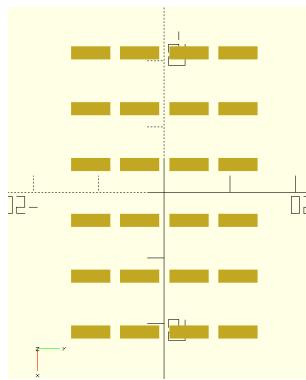


Figura 22.2: Matriz completa de rayos de Sol.

Cecilia contempló ensimismada su texto. Recordó que cada pixel tenía como nombre secreto sus coordenadas, tal como podía apreciar en la figura 22.3(a). Para el caso del dígito '2', por ejemplo, los píxeles que debían crearse eran los mostrados en la figura 22.3(b).

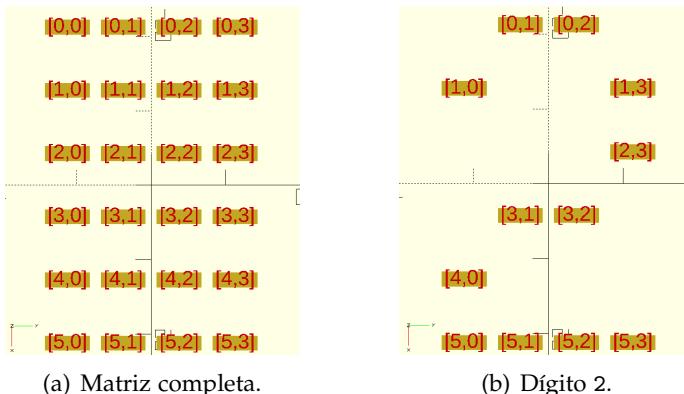


Figura 22.3: Coordenadas de los rayos de Sol que forman un dígito.

22.1. LA MANERA DE CECILIA

Cecilia cruzó los brazos sobre su pecho, frunciendo el ceño y recostándose con cierta frustración contra el respaldo de su silla: no veía cómo plasmar esa idea en palabras. Pero de pronto una luz, sin que supiera bien de dónde, cruzó su mente como un relámpago: «¿Y si paso esas coordenadas como un vector al módulo, y en lugar de pedirle que recorra toda la matriz, le exijo que sólo cree los pixeles cuyas coordenadas forman parte del vector...?». Presa de un súbito entusiasmo, se lanzó sobre el teclado. Luego de unos cuantos minutos, cuya cantidad no hubiera podido precisar, y que incluyeron varias apelaciones a las siempre fieles teclas **[Supr]** y **[←]**, obtuvo lo siguiente:

```

1 dos=[[0,1], [0,2], [1,0], [1,3], [2,3], [3,1],
      [3,2], [4,0], [5,0], [5,1], [5,2], [5,3]];
2
3 module digito(alfa,numero){
4     for(coords=numero){
```

```
5      x=(coords[0]-2.5)*(alto_pixel+delta_alto);  
6      y=(coords[1]-1.5)*(ancho_pixel+delta_ancho);  
7      translate([x,y,-0.01])  
8      rayo_de_sol(alfa);  
9  }  
10 }  
11  
12 digito(90,dos);
```

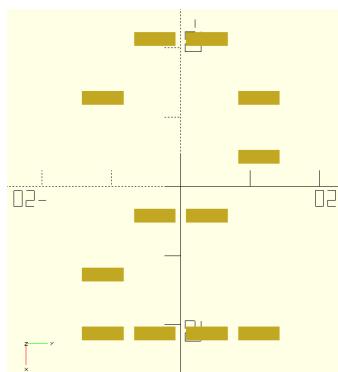


Figura 22.4: Dígito 2 logrado por Cecilia.

Cecilia estuvo a punto de gritar de alegría. Lentamente repasó su texto, con los ojos radiantes de emoción: En la línea 1 definía un vector, cuyos elementos eran las coordenadas de cada uno de los pixeles que formaban el dígito “2”.

El módulo recibía como parámetro ese vector con el nombre `numero`, y en la línea 4 lo recorría asignando cada par de sus coordenadas a la variable `coords`; de esta manera, `coords` valía primero `[0,1]`, después `[0,2]`, luego `[1,0]`, etc.

Para cada valor de `coords` se calculaban en las líneas 5 y 6 un par de `x` e `y`, usando para eso las dos componentes de cada `coords`: `coords[0]` y `coords[1]`. Lo demás, se desprendía de la lógica de su versión anterior del mismo módulo.

Cecilia giró sobre sí misma para mirar a Antonia:

—¿Ves? Ahora sólo resta que escribamos los vectores para cada dígito... ¡Y listo! Ya podremos entonces escribir `digito(angulo,cero)`, `digito(angulo,uno)`, `digito(angulo,dos)`, etc.

Antonia no demostraba la emoción que Cecilia esperaba; de hecho, no mostraba ningún tipo de emoción: sólo miraba atentamente el monitor, con gesto serio y concentrado. Cecilia pensó que en su texto había un *bug* oscuro y secreto, y que Antonia buscaba la manera de revelárselo sin herir su sensibilidad.

Antonia parecióemerger lentamente de sus pensamientos. Volviendo la mirada en dirección a Cecilia, pronunció suavemente:

—Muy bien. Está muy bien resuelto, de hecho. No es la forma en que lo había hecho yo. Te confieso que me sorprende que hubiera otra manera.

Cecilia sintió que la atravesaba una especie de escalofrío: ¿Había encontrado otra forma? Inmediatamente reconoció la naturaleza de la sensación que recorría su cuerpo: era el delicioso vértigo de la independencia; el descubrimiento de que una era capaz de avanzar sin la necesidad de ayuda. No era la primera vez que le ocurría: sintió lo mismo cuando advirtió que su trabajo sobre los espectros estelares cobraba finalmente forma, y supuso que la misma sensación la embargó cuando se largó a dar sus primeros pasos, abandonando los brazos de su madre.

—¿Me contás cómo lo hiciste vos? —preguntó Cecilia con la misma suavidad.

—Por supuesto —respondió con una sonrisa Antonia, que ahora parecía mirarla con otros ojos—. Primero dejame aprovechar para comentarte acerca de una manera alternativa de referirse a los elementos de un vector. Dado que muchas veces se usan para indicar coordenadas —tal fue tu caso, de hecho—, los creadores de OpenSCAD decidieron incluir la posibilidad de que los tres primeros elementos de un vector puedan llamarse usando la notación '`vector.x`', '`vector.y`' y '`vector.z`':

```
1 dos=[[0,1], [0,2], [1,0], [1,3], [2,3], [3,1],
      [3,2], [4,0], [5,0], [5,1], [5,2], [5,3]];
2
3 module digito(alfa,numero){
4     for(coords=numero){
5         x=(coords.x-2.5)*(alto_pixel+delta_alto);
6         y=(coords.y-1.5)*(ancho_pixel+delta_ancho);
7         translate([x,y,-0.01])
8             rayo_de_sol(alfa);
9     }
10 }
11 digito(90,dos);
```

»O sea, coords[0] equivale a coords.x y coords[1] a coords.y; no es más que un poco de ‘azúcar sintáctico’, como suele decirse en el ámbito de este género literario que es la programación... pero a veces resulta expresivo —reconoció Antonia, encogiéndose de hombros.

—Me gusta —dijo Cecilia, que estaba de muy buen humor.

22.2. LA MANERA DE ANTONIA

—Te cuento cómo lo pensé yo —comenzó Antonia—. En mi mente, visualizaba los dígitos como pixeles apagados y encendidos. Entonces se me ocurrió replicar esa imagen mental con una suerte de ‘copia’ matricial, donde cada pixel encendido estuviera representado por un ‘1’ y los demás con un ‘0’:

```
1 dos=[[0, 1, 1, 0],
2      [1, 0, 0, 1],
3      [0, 0, 0, 1],
4      [0, 1, 1, 0],
5      [1, 0, 0, 0],
6      [1, 1, 1, 1]];
```

»La elección de los números 0 y 1 es caprichosa; supongo que como programadora me siento más inclinada por todo lo que parezca binario —reconoció Antonia, y continuó—: Ahora bien, al momento de recorrer filas y columnas, me fijaba si el correspondiente elemento de la matriz era igual a '1': en caso afirmativo, creaba el pixel; si no, no:

```

1 module digito(alfa,numero){
2   for(i=[0:5],j=[0:3]){
3     if(numero[i][j]==1){
4       x=(i-2.5)*(alto_pixel+delta_alto);
5       y=(j-1.5)*(ancho_pixel+delta_ancho);
6       translate([x,y,-0.01])
7       rayo_de_sol(alfa);
8     }
9   }
10 }
11 digito(90,dos);

```

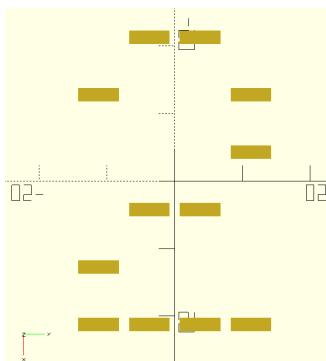


Figura 22.5: Dígito '2' conseguido por Antonia. Como puede comprobarse, el resultado coincide con el conquistado por Cecilia.

Cecilia estudió con atención el texto de Antonia. Tenía mucho sentido: el bucle de la línea 2 recorría, al igual que en su texto original, las filas y columnas completas de la matriz. Pero en la

línea 3 se indagaba si el pixel actual estaba “encendido” en la matriz que expresaba la imagen mental de Antonia: sólo en caso afirmativo se creaba el pixel en las líneas 4 a 7.

—Lo que me gusta de esta forma, si bien está mal que lo diga yo, supongo —admitió Antonia—, es que la forma del dígito está ‘a la vista’ en el texto, bajo la forma de unos y ceros en la matriz: si no te gusta cómo queda, podés cambiarlos fácilmente.

Cecilia expresó su acuerdo con una sonrisa y asintiendo con la cabeza:

—Ahora se trata de escribir una matriz por dígito, ¿no?

—Más o menos —Antonia guiñó un ojo a Cecilia, intentando parecer piola—. Se me ocurrió meter todos los dígitos en un vector..: ¡Un vector de matrices, je!¹

```
1 digitos = [
2 [[0, 1 ,1, 0], // cero
3 [1, 0 ,0, 1],
4 [1, 0 ,1, 1],
5 [1, 1 ,0, 1],
6 [1, 0 ,0, 1],
7 [0, 1 ,1, 0]],
8 [[0, 0 ,1 ,0], // uno
9 [0, 1 ,1, 0],
10 [0, 0 ,1, 0],
11 [0, 0 ,1, 0],
12 [0, 0 ,1, 0],
13 [0, 1 ,1, 1]],
14 [[0, 1 ,1, 0], // dos
15 [1, 0 ,0, 1],
16 [0, 0 ,0, 1],
17 [0, 1 ,1, 0],
18 [1, 0 ,0, 0],
19 [1, 1 ,1, 1]],
```

¹El vector completo, junto con el texto del reloj a medida que Cecilia y Antonia lo conquistan, pueden encontrarse en <https://github.com/lopezsolerluis/reloj-de-sol-digital>.

```
20 [[0, 1 ,1, 0], // tres
21 [1, 0, 0, 1],
22 [0, 0, 1, 1],
23 [0, 0, 0, 1],
24 [1, 0, 0, 1],
25 [0, 1, 1, 0]],
26 // etc....
```

»Lo que gano con esto es que el dígito ‘2’ va a llamarse, dentro del texto, ‘dígitos[2]’, y no ‘dos’ —indicó Antonia, y ante la mirada de Cecilia, que expresaba inequívocamente que no alcanzaba a apreciar el alcance de esa sutil diferencia, agregó—: El problema es que ‘dos’ es una palabra, y por lo tanto menos dúctil a la hora de ser escogida mediante un algoritmo.

Antonia se dio cuenta de que, en lugar de aclarar, estaba oscureciendo la cuestión; se removió con impaciencia en su silla, como cada vez que se sentía ineficaz para explicar algo:

—La idea es que, en breve, deberemos escribir una manera de elegir, en cada momento, qué dígito crear y con qué ángulo, dependiendo de la hora del día. Y los algoritmos producen con más facilidad números que palabras. Así que será mejor que nuestro módulo reciba un número, en lugar de una palabra; mirá:

```
1 module digito(alfa,numero){
2     for(i=[0:5],j=[0:3]){
3         digito=digitos[numero];
4         if(digito[i][j]==1){
5             x=(i-2.5)*(alto_pixel+delta_alto);
6             y=(j-1.5)*(ancho_pixel+delta_ancho);
7             translate([x,y,-0.01])
8                 rayo_de_sol(alfa);
9         }
10    }
11 }
12 digito(90,2);
```

»En la línea 3 rescato el dígito correspondiente del vector de los dígitos. Lo demás quedó igual. Pero fijate que en la línea 12 ahora puedo invocar el módulo indicando simplemente el número 2.

Cecilia expresó su comprensión con una sonrisa admirativa; Antonia pareció entusiasmarse:

—Incluso podría omitir el paso de la creación de la variable `digito`:

```

1 module digito(alfa ,numero){
2     for(i=[0:5] ,j=[0:3]){
3         if(digitos [numero] [i] [j]==1){
4             x=(i-2.5)*(alto_pixel+delta_alto);
5             y=(j-1.5)*(ancho_pixel+delta_ancho);
6             translate([x,y,-0.01])
7                 rayo_de_sol(alfa);
8         }
9     }
10 }
```

Cecilia no se sentía capaz de decidir cuál de las dos últimas versiones le parecía mejor; en cualquier caso, le urgía la necesidad de comprobar si funcionaban con cualesquiera dígitos y ángulos.

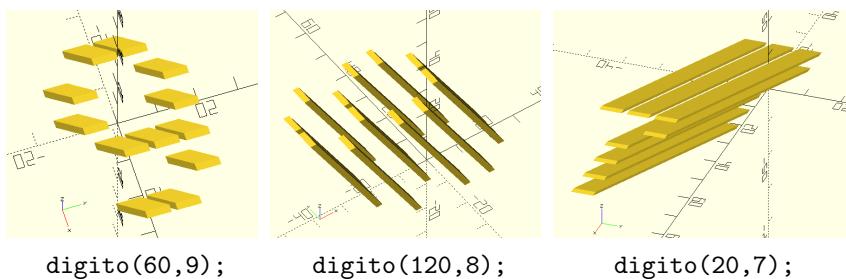


Figura 22.6: Cecilia comprueba la corrección del módulo `digito`.

La figura 22.6 parecía asegurarle que sí.

Un poquito (casi nada) de Astronomía

C^ECILIA Y ANTONIA caminaban morosamente por los jardines de Harvard. El otoño se encontraba avanzado, pero el Sol de la tarde alcanzaba para acariciarlas suavemente. Hojas secas, doradas y carmesíes, danzaban capichosamente a su paso al aire impuesto por una brisa imprevista y veleidosa. Ensimismadas en sus pensamientos, compartían el delicado e íntimo don del silencio Enriquecido y poblado por el afecto.

Cecilia se detuvo con suavidad. Cerrando los párpados, levantó el rostro al cielo para sentir los rayos de Sol filtrándose entre las ramas de los árboles añosos de Harvard. La leve caricia solar le recordó el comienzo de esta aventura; abriendo los ojos dirigió la mirada a su amiga:

—Antonia, creo que es momento de que discutamos el valor del ángulo que debe recibir el módulo digito a fin de seguir fielmente la dirección del Sol en el cielo.

Antonia asintió con un gesto que tal vez era de resignación. Juntas retomaron su paseo entre las hojas caídas y danzantes, los árboles, la brisa y los rayos de Sol.

23.1. EL GIRO DE LA TIERRA

Antonia recorrió con su mirada el jardín a su alrededor. Parecía a punto de lanzar una proclama al mundo entero; con voz firme, anunció:

—La Tierra gira sobre sí misma, alrededor de una línea imaginaria que llamamos ‘eje del mundo’.

Cecilia abrió grandes los ojos, y llevándose una mano al pecho, exclamó:

—¡No! ¿En serio?

Ambas amigas se permitieron reír levemente: ese tipo de bromas sobreactuadas las divertían mucho.

—Pues sí, doctora; así es —confirmó Antonia, inclinando ligeramente la cabeza—. Pero, claro: la percepción inmediata que se ofrece a nuestros sentidos es muy otra y simétrica: a nosotras, ingenuas observadoras terrestres, nos parece que es el cielo el que gira alrededor nuestro: estrellas, planetas y nebulosas en coordinada y diaria danza. Y de esa coreografía incesante, por supuesto, también participa nuestra querida estrella particular: el Sol.

23.2. EL GIRO APARENTE DEL SOL

Cecilia evocó con nostalgia sus primeras lecciones de Astronomía. Como en toda primera aproximación a una materia nueva, no había aprendido mucho, naturalmente. Además, su profesor inicial había sido Luis López, que garantizaba con sus exposiciones lentas, desorganizadas y ambiguas la confusión y la perplejidad. A pesar de eso y misteriosamente, su entusiasmo por la Astronomía no decayó y pudo conquistarla luego con eficacia y profundidad.

El giro de la Tierra sobre su propio eje se refleja como una rotación aparente del cielo en sentido contrario; aún conservaba entre sus viejos apuntes algunos dibujos que Luis hacía pasar por explicaciones. Previendo que en este capítulo discutirían el tema,

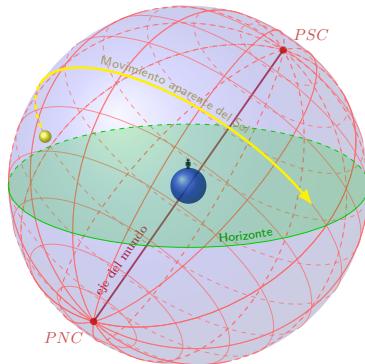


Figura 23.1: Movimiento diurno aparente del Sol, representado sobre el modelo de la esfera celeste. PSC y PNC son los polos celestes sur y norte, respectivamente: proyecciones en el cielo de los homónimos polos terrestres, y extremos del eje alrededor del cual nuestro planeta gira. La Tierra y el observador, naturalmente, se hallan fuera de escala: en comparación con la esfera que representa el cielo, observador y planeta no son más que un ínfimo punto central, inmerso en el plano del horizonte.

consideró conveniente llevar su carpeta de aquel curso al paseo por los jardines. Encontró rápidamente la figura 23.1: en ella se adivinaba un observador parado sobre la Tierra —a la que siente inmóvil bajo sus pies— mientras el cielo (que puede considerarse, con un modelo simplista pero efectivo, como una esfera en cuyo centro se encuentra) gira a su alrededor en el término de un día y de este a oeste —contrariamente al giro real de la Tierra de oeste a este sobre su propio eje.

23.3. LA POSICIÓN DEL RELOJ

—Como recordarás —dijo Antonia mirando el dibujo que Cecilia tenía en sus manos—, el cuerpo de nuestro reloj será un

semicilindro horadado de manera tal de transformar la luz del Sol en rayos discretos que configuren los dígitos que indicarán la hora actual debajo del mismo.

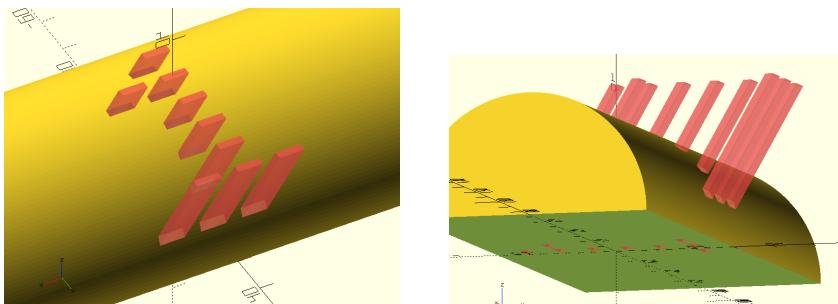


Figura 23.2: Los orificios en el reloj deben permitir que los rayos de Sol indiquen la hora correcta.

Antonia se detuvo, mientras parecía buscar la mejor manera de proseguir:

—Nosotras, entonces, queremos que el Sol, en su movimiento diario, ‘gire’ también alrededor de nuestro reloj. Así, a medida que el ángulo que forma el Sol con la base del semicilindro cambia, cambiarán también los agujeros con que debemos perforarlo.

—En otras palabras —concluyó Cecilia—, debemos colocar el eje del semicilindro paralelamente al eje del mundo, puesto que el Sol parece girar alrededor del mismo.

Antonia aprobó, asintiendo con un movimiento de cabeza:

—El eje del mundo, como seguramente tendrás en algún lugar de tus apuntes, forma con el horizonte un ángulo igual a la latitud del lugar donde una se encuentra —Antonia suspiró con hastío—. ¡A Luis le encanta demostrarlo!

Cecilia lo confirmó alzando los ojos, con un gesto de cansancio sobreactuado.

Antonia se dirigió con paso ligero a un banco cercano; en los suyos flotaba ahora una expresión divertida.

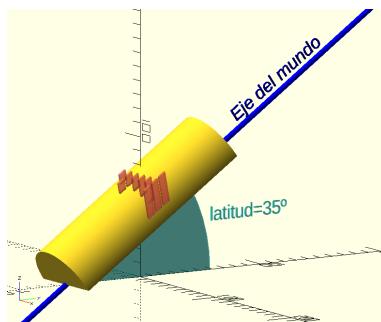


Figura 23.3: El cuerpo del reloj debe ser ubicado paralelamente al eje de la Tierra, puesto que queremos que el Sol gire, de manera aparente, alrededor del mismo.

—A mí me gustan dos casos particulares —dijo, mientras se recostaba boca arriba sobre el largo banco de madera—. Si estás ubicada en un punto cualquiera de la línea del Ecuador, el eje del mundo yace en el horizonte, y los astros se mueven entonces aparentemente de manera perpendicular a ese plano.

Antonia movía los brazos de manera perpendicular al piso, describiendo círculos tan amplios como su sonrisa.

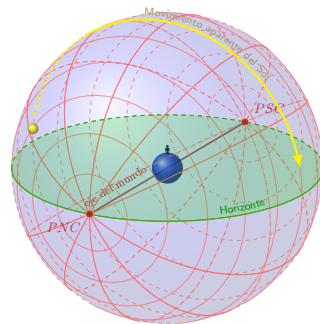


Figura 23.4: Movimiento diurno aparente del Sol, tal como es apreciado por un observador ubicado en el Ecuador terrestre.

Cecilia, sin saber muy bien porqué, extendió también los suyos, pero horizontalmente hacia los costados mientras, de pie junto al banco, comenzaba a girar lentamente sobre sí misma:

—Mientras que si te hallaras en uno de los polos de la Tierra, el eje del mundo sería perpendicular a tu horizonte y las estrellas girarían así a tu alrededor.

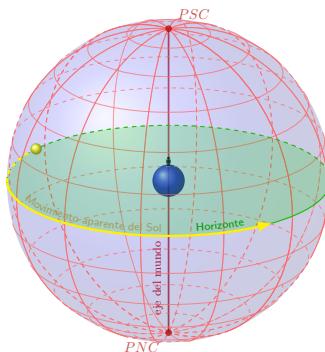


Figura 23.5: Movimiento diurno aparente del Sol, tal como es apreciado por un observador ubicado en el polo sur terrestre.

Cecilia, tras varias vueltas sobre sí misma, simuló con suma gracia que sentía un mareo; su cuerpo se bamboleó exageradamente a un lado y otro, y se desplomó sobre el banco en el que Antonia, riendo, ya se había incorporado.

Cuando el eco de sus risas se apagó, confundiéndose con el rumor de las hojas que seguían bailando con la brisa y los rayos de Sol, Cecilia juzgó apropiado afrontar la cuestión del ángulo:

—¿Y, Antonia? Ya sabemos cómo orientar nuestro reloj, pero ¿qué hacemos con el dichoso ángulo para formar los dígitos?

En lugar de la respuesta esperada, encontró que Antonia estaba contemplando ahora las figuras 23.1, 23.4 y 23.5 con gesto de desaprobación:

—Ahora vamos, Cecilia —protestó—; ¡pero no me digas que

Luis les contó que el Sol se encuentra siempre equidistante de ambos polos celestes!

—¡Por supuesto que no! —respondió Cecilia, riendo—. ¡El curso no era *tan* malo! Alcanzamos a ver que, debido a la inclinación del eje terrestre, el Sol durante el año se acerca alternativamente al polo norte y al polo sur celestes, separándose del plano del ecuador celeste¹ hasta un máximo de unos $23,5^\circ$.

Tras unos instantes, encontró la prueba:

—¿Ves? Acá tengo las figuras 23.6 y 23.7: en ambas podés ver el Sol surcando el cielo ora más cerca del polo sur celeste, ora más cerca del polo opuesto.

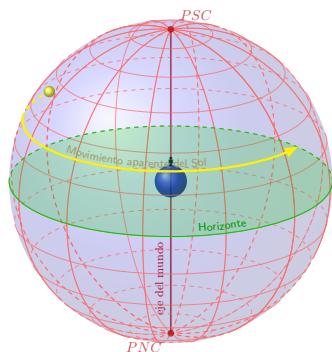


Figura 23.6: Movimiento diurno aparente del Sol, visto desde el polo sur terrestre el 21 de diciembre.

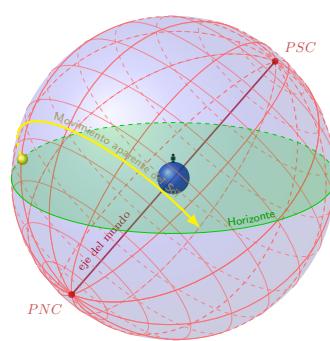


Figura 23.7: Movimiento diurno aparente del Sol, visto desde una latitud de 35°S el 21 de junio.

—Bien —Antonia aprobó displicentemente—. En cualquier caso, el movimiento aparente del Sol, sin importar qué tan cerca se encuentre de cada polo celeste, discurre diariamente en un plano perpendicular al eje del mundo, o sea, en un *paralelo celeste*;

¹Llámase *ecuador celeste* al plano perpendicular al eje del mundo y equidistante de ambos polos celestes.

así que nuestra conclusión anterior acerca de la orientación del reloj no cambia: debemos colocarlo paralelamente al eje terrestre.

Cecilia pareció ofuscarse ligeramente:

—Nunca discutí eso, creo.

—No, está bien —insistió Antonia—; sólo lo aclaraba.

—Pues no hacía falta —replicó Cecilia, contrariada.

23.4. EL DICHOSO ÁNGULO

Levantándose del banco, retomaron su lento paseo en silencio. Tras unos instantes, Antonia creyó conveniente retomar la cuestión del ángulo:

—Podemos llamar *día solar* a una rotación aparente completa del Sol alrededor nuestro. Resulta cómodo dividirlo en 24 partes, dado que ese número tiene muchos divisores.

—Es la misma razón por la que las medialunas se venden por docena: la capacidad del número 12 para dividirse en partes enteras promueve la concordia de familias y grupos de amigos —agregó Cecilia—. Bueno, salvo que los comensales sean cinco o diez...

—Por esa razón es tan popular el número 60: tiene los mismos divisores que 12, pero también los de 10. Aunque creo que una ‘sesentena’ de medialunas excedería el presupuesto de casi cualquier familia —admitió Antonia—. En cualquier caso, a esas 24 partes las llamamos *horas*.

—Y de esas 24, decidimos que la hora 12 coincida con la posición más alta que adquiere el Sol en su movimiento aparente, y la hora 0 con la opuesta —añadió Cecilia.

—Exacto —aprobó Antonia—. De hecho, a esas posiciones del Sol las llamamos *culminación superior* y *culminación inferior*, respectivamente.

Cecilia sonrió maliciosamente:

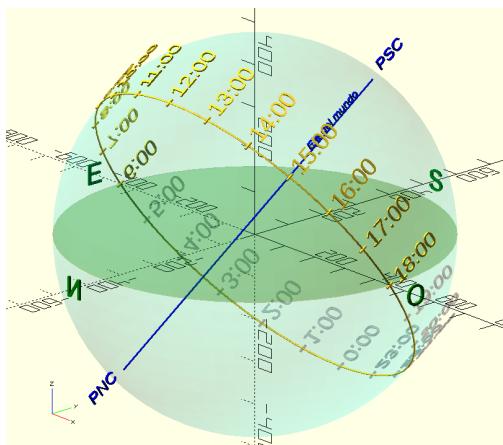


Figura 23.8: El ecuador celeste dividido según las horas ocupadas por el Sol en su movimiento aparente diurno.

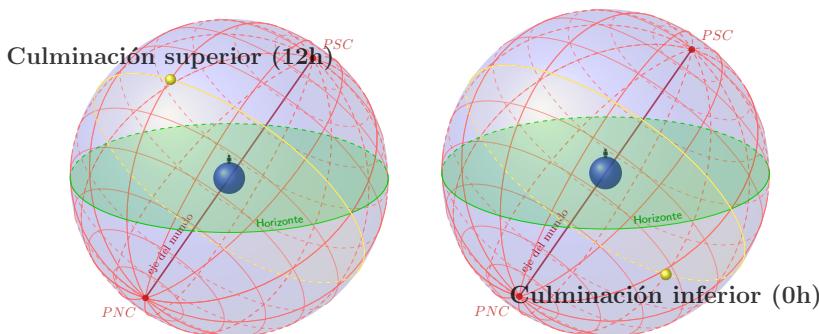


Figura 23.9: El Sol en sus culminaciones superior e inferior.

—Claro —dijo, procurando que su tono de voz no sonara demasiado picante, a la vez que buscaba en su carpeta—. Mirá; acá están: en la figura 23.9 se aprecian ambas culminaciones.

Antonia miró fijamente a Cecilia:

—Está bien —admitió—; parece que algunas cosas vieron en

ese curso. En todo caso —prosiguió, acelerando un poco el paso—, creo que ya estamos en condiciones de determinar, para cada hora solar, el ángulo que los rayos de Sol formarán con el semicilindro de nuestro reloj.

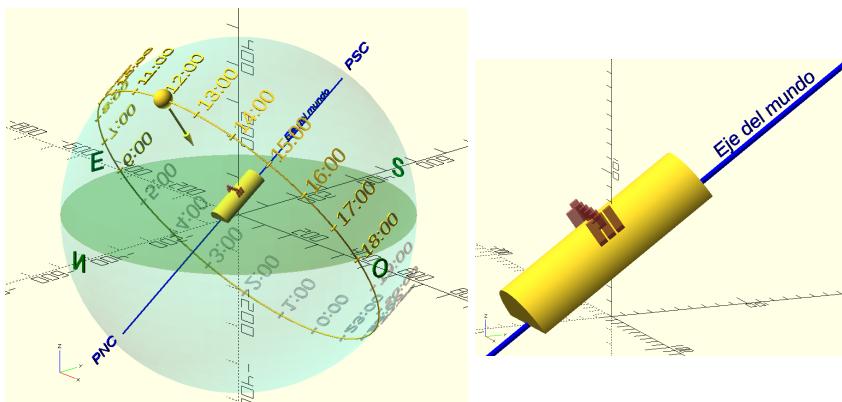


Figura 23.10: A las 12 horas, el ángulo debe ser 90° .

—¡Sí! —afirmó Cecilia, con entusiasmo—. A las 12 horas, el ángulo debe ser de 90° .

—Y a las 9 horas, 135° —se apresuró Antonia a agregar con una sonrisa.

—¡Y a las 15 horas, 45° ! ¡Canté! —gritó Cecilia, riendo.

—Perfectamente, entonces —Antonia recapituló—; está claro que el ángulo variará desde 180° hasta 0° conforme las horas avanzan desde las 6:00 hasta las 18:00.

Cecilia sólo pudo expresar su acuerdo con una sonrisa: ¿estaría emocionada por encontrarse más cerca de concretar el reloj? ¿O sólo cansada de pensar y caminar? No lo sabía, y lo cierto es que poco le importaba: en cualquier caso, la sensación que la embargaba ahora se parecía mucho a la felicidad.

—Cuando volvamos a la oficina veremos cómo escribir matemáticamente esa relación entre horas y ángulos que acabamos

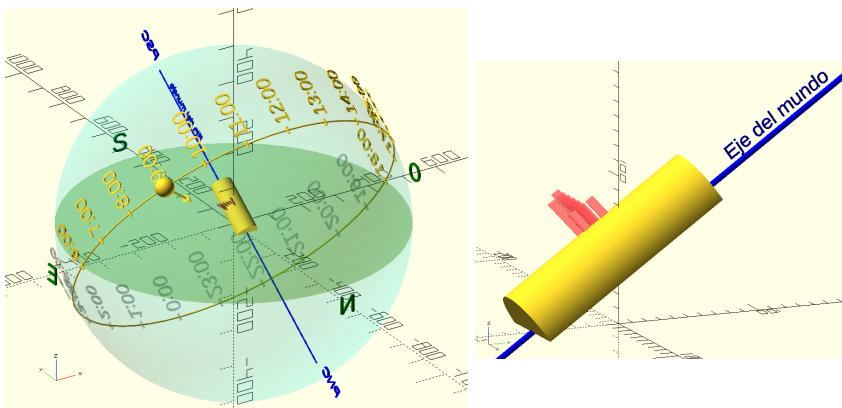


Figura 23.11: A las 9 horas, el ángulo debe ser 135° .

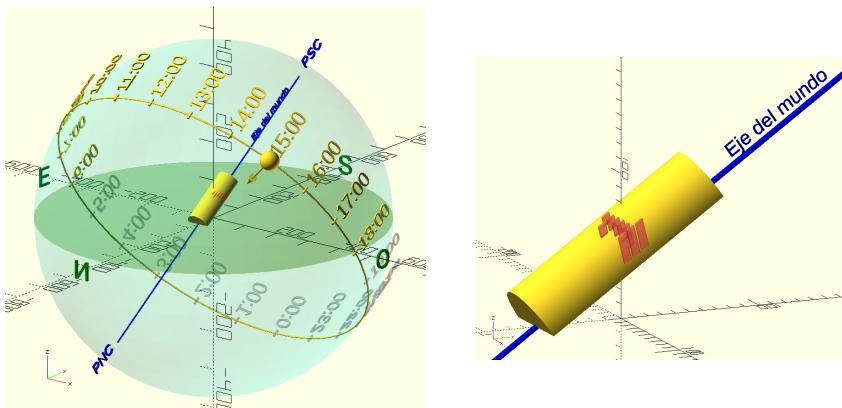


Figura 23.12: A las 15 horas, el ángulo debe ser 45° .

de establecer conceptualmente —anticipó Antonia, que también parecía querer concluir el capítulo—. Pero me gustaría que discutíramos un detalle más. Fíjate que establecimos esa relación para

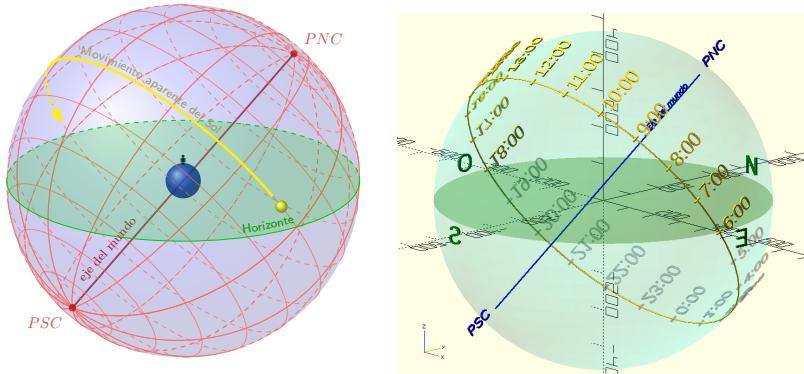


Figura 23.13: Movimiento diurno aparente del Sol, tal como es apreciado por un observador ubicado en el hemisferio norte terrestre.

un sitio como Harvard, cuya latitud es austral^{2,3}: ¿Será la misma para un sitio del hemisferio norte?

Cecilia se detuvo un instante antes de retomar el paso; efectivamente, la cuestión era interesante en sí misma, pero además útil: sería muy bonito que el reloj que estaban escribiendo pudiera concretarse en cualquier sitio del planeta. Buscando nuevamente en su carpeta encontró lo que buscaba: en la figura 23.13 podía comprobarse que para el caso del hemisferio norte terrestre el este y el oeste quedaban intercambiados en relación con el cuerpo del reloj; por esa razón, para las latitudes boreales a las 9:00 le corresponde un ángulo de 45° , a las 15:00 uno de 135° , etc.

—No parece un problema muy difícil de resolver —aventuró Cecilia—; supongo que deberemos escribir dos relaciones matemáticas entre horas y ángulos: una para cada hemisferio. Y

²¿Ignora acaso el autor que la célebre universidad de Harvard está ubicada en el hemisferio norte? (Nota del Editor)

³No entendiste nada.

permitir que el usuario indique, mediante una variable, si desea usar su reloj en el norte o en el sur, y en función de eso emplear una u otra relación: suena a una tarea para un `if`.

Antonia sonreía mientras escuchaba y caminaba junto a Cecilia. Pensó que ya empezaba a sonar como una programadora: elevando suposiciones al rango de algoritmos, cifrando sentencias breves al borde de lo confuso, y hasta enfrentando problemas aún no resueltos con una confianza demasiado parecida a la pedantería. Por un momento temió estar echándola a perder; pero decidió que, en cualquier caso, ya era demasiado tarde.

Los últimos rayos de Sol conseguían aún filtrarse entre las hojas de los añosos árboles de Harvard. La brisa era más fresca, y el color de las hojas empezaba a desaparecer como desaparecen todos los colores al caer la noche. Cecilia y Antonia, en su paseo, ya estaban llegando a los pies de los muros de Harvard. Algunos astrónomos, cansados de su trabajo diario, se asomaban por las altas ventanas y miraban con un ligero desdén a las dos amigas que habían dedicado la tarde aparentemente a pasear.

Antonia y Cecilia notaron esas miradas acusadoras; sin mediar palabra entre ellas supieron inmediatamente qué debían hacer. Se separaron unos pasos, y Antonia dio comienzo a la danza:

—Cecilia, ¿cómo era el movimiento aparente de las estrellas cuando estamos ubicadas en un polo..?

Y ambas abrieron ceremoniosa y ampliamente los brazos, y ante los escandalizados ojos de los astrónomos de Harvard comenzaron a girar, y a girar, y a girar, mientras bailaban con las hojas y los árboles y la brisa y los últimos rayos de Sol.

Funciones

CECILIA Y ANTONIA volvieron a enfrentarse al día siguiente a la computadora, dócil receptora de sus textos.

—Muy bien —esta vez fue Cecilia quien empezó—; las relaciones matemáticas entre horas y grados para ambos hemisferios no son difíciles de lograr —y tras garrapatear unos instantes sobre uno de los tantos papeles que llenaban de confusión el escritorio de Antonia, presentó dichas funciones:

$$G_{\text{sur}}(H) = 270^\circ - 15^\circ \times H \quad (24.1a)$$

$$G_{\text{norte}}(H) = 15^\circ \times H - 90^\circ \quad (24.1b)$$

»H es el valor de la hora que queremos transformar en grados; $G_{\text{sur}}(H)$ y $G_{\text{norte}}(H)$ son las funciones que realizan dicha transformación para ambos hemisferios —aclaró Cecilia.

—¿Te molesta si compruebo la veracidad de tus relaciones probándolas con algunos valores particulares? —preguntó Antonia procurando con un tono neutro disimular su intención de molestar a su amiga.

Cecilia, por su parte, no pudo esconder su contrariedad:

—Por supuesto que no; dale, adelante —dijo, mientras se recostaba con cierta brusquedad contra el respaldo de la silla.

—A ver... Veamos... —musitó Antonia mientras escribía sobre el mismo pedazo de papel—. Si uso la ecuación 24.1a con las horas 6, 9, 12, 15 y 18, obtengo... —y tras unos momentos en los que aprovechó los últimos rincones del ya atestado papel, confirmó—: Sí, perfecto: me da 180° , 135° , 90° , 45° y 0° , tal como debe ser.^{1,2}

—¿No querés confirmar también la eficacia de la fórmula 24.1b? —soltó Cecilia con una sonrisa quebrada que podía tomarse como un desafío.

—No hace falta —admitió Antonia, satisfecha—. Estoy segura de que está bien. Ahora lo que tenemos que hacer es escribirlas en el lenguaje de OpenSCAD.

24.1. `function`

—Para escribir una función usamos la expresión `function` —dijo Antonia, mientras tomaba el teclado.

```
function suma(a,b)=a+b;
```

»Como podés apreciar, la sintaxis es bastante intuitiva —Antonia, una vez más, confundía deseo con realidad—. La palabra `function` es seguida por el nombre que queremos darle a la función; en este caso ‘suma’. Al nombre le siguen, encerrados entre paréntesis, los parámetros que dicha función admitirá y sobre los que va a operar. A continuación, y tras un signo ‘=’, sigue la expresión que devolverá la función como valor.

A Cecilia no le parecía insensato lo que Antonia le contaba, pero sentía que sólo unos ejemplos le darían la seguridad que necesitaba para entender un giro lingüístico nuevo.

¹El hecho de que la fórmula funcione para cinco valores particulares no garantiza su eficacia para todo el conjunto continuo que abarca las 12 horas que van de las 6:00 a las 18:00. (Nota del Editor)

²Sos cabeza dura, eh! (Nota de Antonia, Cecilia y Luis)

—En la descripción de cualquier nuevo objeto, la invocación a la función vale tanto como el resultado que devuelve; mirá —siguió Antonia:

```
cube([suma(1,1), suma(2,2), suma(-2,3)]);
```

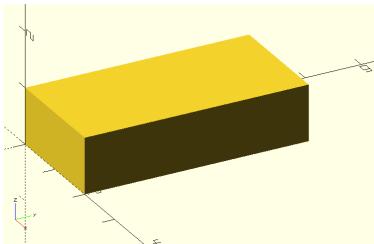


Figura 24.1: Cubo cuyos lados son calculados por una función.

Cecilia pensó que se trataba de una forma muy rebuscada de escribir `cube([2,4,1])`, pero captó la idea; además, ya estaba acostumbrada a los ejemplos triviales de Antonia.

—Por supuesto, este ejemplo es bastante tonto —reconoció su amiga—. Uno que me gusta mucho es éste:

```
1 function elipse_cartesiana(a,e,theta)=
2   let(b=a*sqrt(1-e*e),
3       x=a*cos(theta),
4       y=b*sin(theta))
5   [x,y];
```

Cecilia volvió a sentir en el cuerpo ese vértigo feliz que le producía un misterio en el que se presentía con claridad un orden secreto.

—En una función también podés declarar variables auxiliares —dijo Antonia—, pero debés hacerlo con la palabra clave `let`, que recibe entre paréntesis dichas variables junto con su definición. En esta función en particular, quiero calcular las coordenadas cartesianas x e y del punto de una elipse de la que conozco su semieje mayor (a) y excentricidad (e), para un cierto ángulo θ .

»En las líneas 2 a 4 de la función `elipse_cartesiana` calculo tres variables dentro de un `let`: `b`, `x` e `y`. Las tres, como te conté, deben ir dentro de un par de paréntesis, y separadas unas de otras mediante comas. Finalmente, en la línea 5 indico el resultado que debe devolver la función: el vector `[x,y]`, con las coordenadas buscadas.

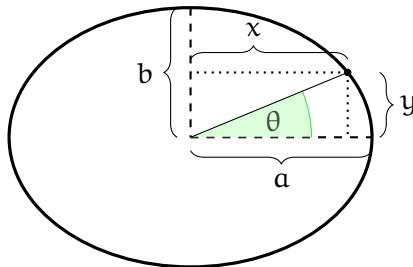


Figura 24.2: Elipse

Cecilia procuró traer a su memoria viejos recuerdos geométricos, para lo cual le resultó muy útil la figura 24.2: para toda elipse se verificaba la siguiente relación entre `a`, `b` (el semieje menor) y `e`:

$$b = a \times (1 - e^2)$$

y `x` e `y` podían expresarse como

$$\begin{cases} x = a \times \cos \theta \\ y = b \times \sin \theta \end{cases}$$

—Una función puede devolver cualquier tipo de valor —explicó Antonia—; ésta en particular devuelve `[x,y]`: un vector cuyos elementos son `x` e `y`. Ahora podemos usarla para escribir, por ejemplo, una elipse de esferas:

```
1 module elipse(a,e,grosor,n_esferas=100){  
2     for(i=[0:n_esferas-1]) {
```

```

3     theta=i*360/n_esferas;
4     pos=elipse_cartesiana(a,e,theta);
5     translate([pos.x,pos.y,0])
6         sphere(r=grosor);
7     }
8 }
9 elipse(100,.7,5);

```

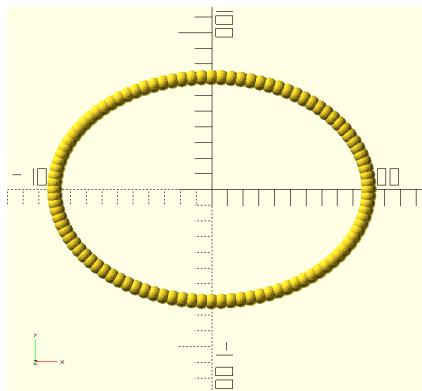


Figura 24.3: Elipse producida por Antonia con una función.

»La línea clave es la 4: en ella se calculan las coordenadas de cada esfera gracias a la función que escribí antes, y se guardan en la variable pos —desarrolló Antonia—. Luego, en la línea 5, rescato esas coordenadas con pos.x y pos.y para efectuar la debida traslación de cada esfera del perímetro de la ellipse.

Cecilia no parecía muy convencida:

—A ver... ¿me dejás probar algo? —y sin esperar respuesta, tomó el teclado para sí.

```

1 module elipse_bis(a,e,grosor,n_esferas=100){
2     for(i=[0:n_esferas-1]){
3         theta=i*360/n_esferas;
4         b=a*sqrt(1-e*e);
5         x=a*cos(theta);

```

```
6      y=b*sin(theta);  
7      translate([x,y,0])  
8          sphere(r=grosor);  
9    }  
10 }  
11 ellipse_bis(100,.7,5);
```

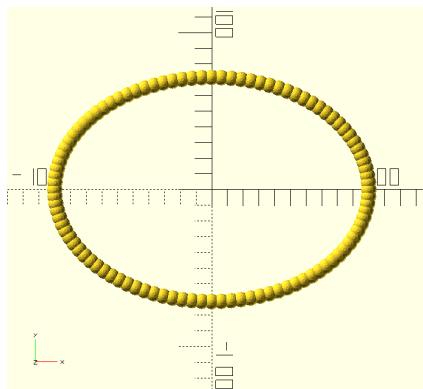


Figura 24.4: Elipse producida por Cecilia sin usar una función.

Cecilia dirigió a Antonia una mirada de interrogación, al tiempo que señalaba el resultado de la figura 24.4:

—¿Qué onda? ¿No se podía hacer así?

—Sí, por supuesto —admitió su amiga—. En principio, puede argumentarse que se trata de una cuestión de gustos. Personalmente, cuando detecto que una serie de cálculos merece ser ‘aislada’, la encierro en una función propia.

Antonia comprendió que como justificación era muy pobre, y hasta injusta respecto de las dilatadas posibilidades expresivas de las funciones. Tras un instante y un inevitable suspiro, trató de mejorarla:

—A medida que una escribe advierte que hay ciertas costumbres o vicios en los que caemos una y otra vez, y que suelen

desencadenar los mismos nefastos resultados: uno de ellos es encargar a un solo bloque de texto (un módulo, por ejemplo) demasiada ‘responsabilidad’ o ‘trabajo’. Cuando eso pasa, y detectás un error luego (a veces, cientos de líneas más tarde), te encontrás sumergida en ese viejo bloque de texto tratando de encontrar en qué línea está el problema. Pero si en lugar de eso vos repartís el trabajo en bloques mínimos, responsables cada uno de una tarea elemental y bien definida, cualquier revisión posterior (¡e inevitable!) será sensiblemente más fácil.

A Cecilia le pareció razonable lo que le confió Antonia, aunque no terminaba de comprenderlo del todo. Pensó que quizá lo mejor sería intentar seguir su consejo, pero sin adoptarlo como una regla rígida que la entumeciera en su proceso de escritura.

—Además —abundó Antonia—, imaginate que más tarde te das cuenta de que hay una forma más eficiente o rápida de calcular el mismo valor: ¡deberías modificar cada aparición de esas fórmulas en cada módulo que las empleara! —Antonia abrió grandes los ojos y separó sus brazos, como si los gestos constituyeran un argumento: otro defecto más de su condición docente. A Cecilia esta razón le gustó un poco más; en cualquier caso, y para no seguir tolerando otra de las interminables apologías de su amiga, decidió lanzarse a la escritura de...

24.2. LAS FÓRMULAS DEL ÁNGULO

—Entonces nuestras fórmulas podrían quedar... ¿así? —preguntó Cecilia, mientras escribía rápidamente.

```
1 function alfa_sur(hora)=270-15*hora;  
2 function alfa_norte(hora)=15*hora-90;
```

—Exacto —confirmó Antonia con una sonrisa.

24.3. echo

—Yo, cuando escribo, no puedo liberarme de una incómoda sensación —confesó Antonia—: aún cuando me encuentro segura de que lo que acabo de escribir es correcto, necesito verificarlo con ejemplos concretos. ¡Ay! ¡Si supieras la de sorpresas que me deparó el uso incomprobado de algunos de mis párrafos..!

Cecilia pensó que Antonia había caído en otra de sus exageraciones retóricas; pero la dejó continuar.

—Una sentencia que me resulta muy útil a fin de conjurar esas contrariedades es `echo`: imprime en la consola lo que recibe entre paréntesis —y Antonia tecleó con rapidez algunos ejemplos tontos.

```
1 echo(1);
2 echo(1+1);
3 echo("Antonia y Cecilia son", 18/9, "amigas
    inseparables.");
```

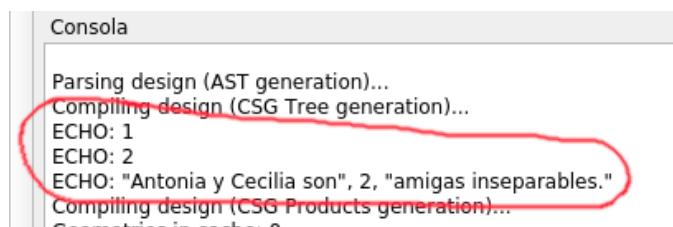


Figura 24.5: Antonia ilustra el uso de la instrucción `echo`.

»Ahora podemos confirmar que nuestras funciones funcionan.

—Antonia no podía evitar esos juegos de palabra niñoños.

```
1 for(h=[6:3:18])
2   echo(h, alfa_sur(h), alfa_norte(h));
```

Cecilia se irguió en su silla, entusiasmada:

—¡Está bueno! —tuvo que reconocer ante la figura 24.6.

```

Consola
Compiling design (CSG Tree generation)...
ECHO: 6, 180, 0
ECHO: 9, 135, 45
ECHO: 12, 90, 90
ECHO: 15, 45, 135
ECHO: 18, 0, 180
Compiling design (CSG Products generation)

```

Figura 24.6: Antonia comprueba las funciones que calculan el ángulo α .

24.4. CONDICIONALES EN FUNCIONES

—Ahora ya podríamos usar esas funciones dentro de nuestros generadores de dígitos, para darles la dirección adecuada en función de la hora y el hemisferio en el que el usuario se encuentra.

—Antonia se detuvo, suspendiendo el último punto en el aire.

Cecilia entendió que eso y el uso del modo potencial estaban sugiriendo que ésa no era la solución que su amiga consideraba óptima. Decidió seguirle el juego:

—Peerooo.... —arrastró, invitándola a seguir.

Antonia sonrió al ver adivinada su intención:

—¿No sería más lindo que el módulo `digito` simplemente confiara en el resultado que le brinde una suerte de función `alfa(H)`, la cual estaría a cargo de decidir qué ángulo ofrecer dependiendo del hemisferio..? ¿Acaso vamos a agregar a las obligaciones de `digito` la de indagar qué función usar..?

Cecilia ya estaba demasiado cansada como para disentir. Además, debía admitir que la idea, en principio, sonaba bien; con un movimiento de cabeza, asintió.

—Bien —aprobó Antonia con una sonrisa, aparentemente satisfecha de su capacidad de persuasión—. Entonces vamos a crear una variable con la que el usuario pueda definir el hemisferio para el cual quiere construir su reloj, y una función `alfa(H)` que utilice `alfa_sur(H)` o a `alfa_norte(H)` dependiendo del valor de

24. FUNCIONES

aquella —y con entusiasmo volvió a sacar ligeros chasquidos del ya fatigado teclado.

```
1 hemisferio="sur";
2
3 function alfa(hora)=
4     hemisferio=="sur" ? alfa_sur(hora) :
                           alfa_norte(hora);
```

Cecilia volvió a erguirse en su silla:

—¡Epa! —exclamó—. ¿Qué es eso?

—Es la forma en que se escriben las estructuras condicionales en las funciones —declaró Antonia, quien disfrutaba sensiblemente provocando la perplejidad en su amiga—. En las mismas no se puede emplear la sentencia `if`. No me preguntes —se atajó, levantando una mano—; ignoro porqué. Lo cierto es que si queremos responder a una condición dentro de una función debemos apelar al operador ternario '`? :`'. A ver con un ejemplo:

```
1 function inversa(x)= (x==0) ? "0 no tiene
                                inverso" : (1/x);
2
3 echo(inversa(10));
4 echo(inversa(0));
```

Consola

```
Compiling design (CSG Tree generation)...
```

```
ECHO: 0.1
```

```
ECHO: "0 no tiene inverso"
```

```
Compiling design (CSG Products generation)
```

Figura 24.7: Empleo del operador ternario '`? :`'.

Antonia tomo aire, como cada vez que se veía ante la perspectiva de tener que ofrecer una larga explicación:

—La idea es ésta: el operador '`? :`' espera tres expresiones. La primera debe adoptar la forma de una condición (en el último ejemplo, se trata de `(x==0)`). Si la condición es verdadera,

el operador devuelve el valor de la segunda expresión (el texto "0 no tiene inverso" en nuestro ejemplo); en caso contrario, devuelve el valor de la tercera expresión ($(1/x)$ en nuestro caso).

—¿Es como un `if/else`, pero para funciones? —preguntó Cecilia, más por decir algo que para confirmar que había entendido, cosa de la que no estaba tan segura.

—¡Exacto! —exclamó Antonia con entusiasmo, queriendo creer que su explicación había sido eficaz—. Te comento que los paréntesis en cada expresión no son necesarios, salvo cuando se imponen para resolver alguna ambigüedad. El ejemplo anterior podría haberse escrito tranquilamente así:

```
function inversa(x)= x==0 ? "0 no tiene inverso"
: 1/x;
```

»¡Ah! Una cosa importante —Antonia sobreactuó como era su costumbre al comentar los detalles finales de una novedad—: Las dos últimas expresiones pueden ser de cualquier tipo, con tal que representen un valor: un número, un texto entre comillas, un vector, una matriz, etc.; cualquier cosa que vos puedas asignar a una variable.

Cecilia volvió ahora sus ojos a la definición de la función `alfa(hora)`. No le pareció tan difícil de entender, finalmente: el operador ternario primero indagaba la veracidad de la expresión `hemisferio=="sur"`. Si era verdadera, devolvía el valor de la función `alfa_sur(hora)`; en caso contrario, retornaba el de `alfa_norte(hora)`. Sonaba bien, pero ahora fue ella quien quiso comprobar su funcionamiento:

```
1 hemisferio="sur";
2 for(h=[6:3:18])
3   echo(h, alfa(h));

1 hemisferio="norte";
2 for(h=[6:3:18])
3   echo(h, alfa(h));
```

24. FUNCIONES

```
Consola
Compiling design (CSG Tree generation)...
ECHO: 6, 180
ECHO: 9, 135
ECHO: 12, 90
ECHO: 15, 45
ECHO: 18, 0
Compiling design (CSG Products generation)
```

Figura 24.8: Calculo del ángulo α para el hemisferio sur.

```
Consola
Compiling design (CSG Tree generation)...
ECHO: 6, 0
ECHO: 9, 45
ECHO: 12, 90
ECHO: 15, 135
ECHO: 18, 180
Compiling design (CSG Products generation)
```

Figura 24.9: Calculo del ángulo α para el hemisferio norte.

Parecía que la función funcionaba.

24.5. UNA ANTIGUA FUNCIÓN

Cecilia contemplaba feliz las funciones conquistadas, sin saber si adjudicar esa felicidad al sentido estético de tan altas construcciones sintácticas o al presentimiento de que al fin estaban más cerca de lograr su tan ansiado reloj de Sol digital. Pero en medio de su dicha, sintió que se colaba cierta sensación de nostalgia: una especie de recuerdo difuso de algo que no había cerrado por completo.

—¡Antonia! —exclamó de repente—. Esta construcción de una función, junto con los signos ‘?’ y ‘:’ ya la vimos, ¿no? —y dirigió a su amiga una mirada en la que brillaba un pedido de ayuda.

—Claro —dijo suavemente Antonia, con una amplia sonrisa—. ¡Qué memoria! Pensé que te habrías olvidado. Fue hace mucho,

en el capítulo 8:

```

1 radio=16;
2 factor=1.2;
3
4 function z(i,acc=0) =
5   (i==0)
6     ? radio*acc
7     : z(i-1,acc=acc+(factor+1)/pow(factor,i));

```

Cecilia sintió que un relámpago cruzaba su mente:

—¡Sí! ¡Ahora lo recuerdo bien! Era la solución que casi no me mostrás en ese momento... —confirmó, devolviendo a Antonia la sonrisa y rememorando ese ya viejo capítulo—. «¡Cuánto avanzamos desde entonces..!» —pensó, con una suave melancolía.

Cecilia recorrió ahora el texto con la seguridad que le infundía su nueva conquista de las funciones y sus condicionales: En primer lugar, se verificaba si *i* era nulo en la línea 5. En caso afirmativo, se devolvía el producto de *radio* y *acc*. En caso contrario... Cecilia detuvo su lectura con estupor. «¡Eh! ¡Qué es eso!» —pensó con alarmada perplejidad—. «¿En caso negativo debe devolverse el valor de la propia función *z*? ¡Pero si estamos dentro de la función *z*!».

El gesto de Cecilia debió haber sido lo suficientemente elocuente por sí mismo, ya que Antonia se sintió obligada a intervenir:

—Se trata de un caso muy usual en este género literario: es un ejemplo de *recursividad*. La recursividad es muy común también en ese otro gran género literario, tan emparentado con la programación: la matemática. Recordemos, por ejemplo, la definición más difundida de la función factorial:

$$n! = \begin{cases} 1 & \text{si } n = 1, \\ n \times (n - 1)! & \text{si } n > 1. \end{cases}$$

Cecilia, por supuesto, recordaba esa definición, la cual tenía un

sentido muy claro: el cálculo de $5!$, por ejemplo, podía pensarse así: como $5 > 1$, $5! = 5 \times 4!$. Pero $4! = 4 \times 3!$, así que $5! = 5 \times 4 \times 3!$. Y así siguiendo, se llegaba hasta $5! = 5 \times 4 \times 3 \times 2 \times 1$, puesto que $1! = 1$, y el descenso a través de los números naturales allí se detenía.

—Esa definición puede escribirse de manera casi literal en OpenSCAD —aseguró Antonia:

```
function factorial(n)= n==1 ? 1 :  
    n*factorial(n-1);
```

»Si $n==1$, devuelve el valor 1; en caso contrario, devuelve el producto $n*factorial(n-1)$. Parece magia... pero funciona —prometió Antonia, mientras escribía la necesaria comprobación:

```
1 for(i=[1:10])  
2     echo(i,factorial(i));
```

The screenshot shows a terminal window titled "Consola". It displays the command "Compiling design (CSG Tree generation)...". Below this, the output of the "echo" command is shown, listing the factorial values from 1 to 10. The output is as follows:

```
ECHO: 1, 1  
ECHO: 2, 2  
ECHO: 3, 6  
ECHO: 4, 24  
ECHO: 5, 120  
ECHO: 6, 720  
ECHO: 7, 5040  
ECHO: 8, 40320  
ECHO: 9, 362880  
ECHO: 10, 3.6288e+6
```

At the bottom of the window, it says "Compiling design (CSG Tree generation)".

Figura 24.10: Antonia demuestra el funcionamiento de una definición recursiva de la función factorial.

Cecilia, frente al resultado ofrecido por la figura 24.10, se encontró fascinada con las posibilidades expresivas que este nuevo misterio le ofrecía; decidió, como con tantas otras novedades conceptuales, dedicarle la escritura de varios ejemplos a fin de conquistarla plenamente.

—Un detalle importantísimo que nunca debés olvidar —advirtió Antonia con gesto adusto— es el establecimiento claro del caso base; en la definición de factorial, se trata de $i=1$. Si ese caso no se alcanzara nunca, la recursión no se detendría jamás. Bueno, en realidad en OpenSCAD hay un límite interno impuesto a toda función recursiva, así que tampoco vas a romper nada. Pero en otros lenguajes de programación podrías colgar la computadora.

Cecilia pensó que de todas formas eso ocurría casi todos los días con las computadoras de Havard, sin necesidad de escribir funciones recursivas con errores. Pero aceptó el consejo de Antonia como razonable.

—Si ahora volvemos a la definición de la función z , verás que se adecua fielmente a la expresión matemáticamente clásica:

$$z(i, \text{acc}) = \begin{cases} \text{radio} \times \text{acc} & \text{si } i = 0, \\ z(i - 1, \text{acc} + \frac{\text{factor}+1}{\text{factor}^i}) & \text{si } i > 0. \end{cases}$$

»que a su vez es la fiel expresión recurrente de esta otra:

$$z(i) = \text{radio} \times \left[0 + \frac{k+1}{k^1} + \frac{k+1}{k^2} + \frac{k+1}{k^3} + \dots + \frac{k+1}{k^i} \right]$$

»En la que reemplacé ‘factor’ por ‘ k ’ para que resultara más legible —y Antonia se recostó contra el respaldo de su silla, mirando su texto como una artista que contemplara el efecto general logrado por su última pincelada sobre un retrato que, como todos, no era otra cosa que un autorretrato.^{3,4}

³Aún podría extraerse $k+1$ como factor común. (Nota del Editor)

⁴Pesado! (Nota de Antonia, Cecilia y Luis)

Una hora escrita con Sol – I

—MUY BIEN, ANTONIA —comenzó Cecilia con entusiasmo—. Ya sabemos cómo formar cada dígito y cómo calcular el ángulo con que cada uno de ellos debe cortar el cuerpo del reloj; ahora sólo debemos, para cada hora del día, agrupar los dígitos correspondientes y... ¡ya tenemos reloj de Sol digital!

Cecilia miró a Antonia con una sonrisa radiante, pero encontró en su amiga una expresión taciturna:

—Antonia... ¿pasa algo? —preguntó.

Antonia pareció salir lentamente de su ensimismamiento:

—No, no... para nada —contestó, pero sus manos se mantenían, contrariamente a su costumbre, lejos del teclado.

—Pues no parece —objetó su amiga, frunciendo el ceño—. Ya casi logramos nuestro objetivo... ¿no te pone contenta?

—Bueno... sí, sí, claro —admitió Antonia sin mucha convicción, mientras se incorporaba con cierto esfuerzo en la silla como quien trata de afirmar sus dichos con la postura corporal.

Cecilia adivinó de pronto lo que estaba sucediendo; suavemente preguntó:

—¿No te gusta la idea de terminar el reloj, no?

Antonia sonrió con tristeza, y con un gesto de alivio al verse sorprendida en su paradójica melancolía, suspiró:

—Supongo que soy bastante predecible... Bien, sí: para qué te lo voy negar: estamos a punto de terminar el reloj, y ya estoy extrañando estas reuniones.

Cecilia sonrió con ternura:

—Vamos, Antonia; te conozco: ¡Ya se te va a ocurrir otra cosa tanto o más loca que este reloj! —y ambas rieron con ganas.

Antonia finalmente tomó el teclado con decisión:

—Bien, necesitamos un módulo que reciba la hora que queremos producir. No hace falta pasarle ningún ángulo: el mismo depende precisamente de la hora, así que será calculado por el mismo módulo.

```
1 module hora_solar(hora){  
2  
3 }
```

Antonia se reclinó contra la silla, mirando el texto con la cabeza ligeramente ladeada:

—Hmmm... esto nos permitirá pedir algo tan natural como `hora_solar(12)`, pero ¿cómo hacemos con las 12:30, por ejemplo? No sé si me gusta tanto la expresión `hora_solar(12.5)`... Por otra parte, 12:30 claramente no es un número. Creo que lo mejor es pasar al módulo la hora y los minutos por separado.

```
1 module hora_solar(horas,minutos){  
2  
3 }
```

Cecilia no estaba muy segura de la superioridad de la segunda versión; pero tampoco le parecía peor, así que decidió confiar en la intuición de su amiga.

—Ahora debemos calcular el ángulo correspondiente; nada más fácil —aseguró Antonia, con evidente satisfacción.

```
1 module hora_solar(horas,minutos){  
2     alfa=alfa(horas+minutos/60);  
3 }
```

Cecilia sonrió, divertida: la función `alfa` requería la hora en unidades de horas: nada más simple de conseguir, simplemente sumando los minutos divididos por 60 a las horas.

—Ahora viene la parte más difícil —advirtió Antonia—: debemos obtener los dígitos que forman las horas y los minutos. Por ejemplo, si el módulo es invocado como `hora_solar(12, 30)`, de alguna forma debe poder obtener un ‘1’, un ‘2’, un ‘3’ y un ‘0’. Convendría almacenar cada dígito en una variable.

```
1 module hora_solar(horas,minutos){  
2     alfa=alfa(horas+minutos/60);  
3     hora_decenas= ? ;  
4     hora_unidades= ? ;  
5     minuto_decenas= ? ;  
6     minuto_unidades= ? ;  
7 }
```

Cecilia sintió que podían evitarse un problema:

—¿Y no podemos sencillamente enviar al módulo cada dígito?

Quiero decir, reescribirlo así:

```
1 module hora_solar(hora_decenas , hora_unidades ,  
2                   minuto_decenas , minuto_unidades ){  
3 }
```

Cecilia no había terminado su sugerencia cuando el gesto de su amiga hizo que se arrepintiera.

—¡Ay, Cecilia! —Antonia no disimuló su expresión de desagrado—. ¡Qué definición tan fea! No me imagino pidiendo, ni quisiera a un módulo, que me muestre la hora 1, 2, 3, 0...

—¿La hora 12, 30 te parece mucho mejor? —replicó Cecilia, algo mortificada.

Antonia no pareció captar la mordacidad del comentario de Cecilia:

—Sinceramente, sí: puedo imaginarme invocando este módulo con un bucle que recorra las horas, de 6 a 18, y para cada una de

ellas los minutos, de 0 a 59. Pero ya llegaremos a eso —Antonia parecía ahora apurada por continuar—. Veamos cómo rescatar dígitos de un número; creo que es tarea para una nueva función.

25.1. FUNCIÓN RESTO

—Vamos a necesitar la función que calcula el resto de un cociente —adelantó Antonia.

```
echo (18 % 2 , 19 % 2 , 21 % 3 , 23 % 3) ;
```



Figura 25.1: Función resto.

» $n \% m$ devuelve el resto del cociente $\frac{n}{m}$; de esta forma, $18 \% 2 = 0$, ya que 18 es un número par, mientras que $19 \% 2 = 1$. De la misma manera, $21 \% 3 = 0$ y $23 \% 3 = 2$.

—¡Bien! Ahora ya sabemos distinguir los números pares de los impares —Cecilia seguía un poco molesta, y lo expresaba con ironías no demasiado sutiles. Antonia, según su costumbre, no dio muestras de percibirlo y continuó:

—Nosotras, en nuestro reloj, podemos aprovechar esta función para rescatar dígitos:

```
echo (1 % 10 , 12 % 10 , 123 % 10 , -1234 % 10) ;
```

Cecilia no pudo evitar que un súbito entusiasmo le cambiara el gesto: apreció claramente que el resto de un número dividido por 10 le devolvía las unidades del mismo... ¡incluyendo su signo!

Antonia sonrió al comprobar el efecto que la aplicación de la nueva función produjo en Cecilia:

```

Consola
Compiling design (CSG Tree generation)...
ECHO: 1, 2, 3, -4
Compiling design (CSG Products generation)

```

Figura 25.2: El resto de la división por 10 rescata las unidades de un número, incluido el signo.

—Bien, ya sabemos cómo rescatar las unidades... ¿Cómo hacemos ahora con las decenas? Y, ya que estamos, con cualquier otra posición: sería una pena no escribir una función bien general, que merezca la aprobación hasta del exigente editor de este manualcito.^{1,2} Pienso en algo como una función `n_a_dígito(n, p)`, donde `n` es el número a diseccionar y `p` la posición del dígito que queremos obtener, contando desde las unidades; por ejemplo, que `n_a_dígito(-1837, 0)` diera `-7` y `n_a_dígito(1986, 2)` devolviera `9`.

Cecilia encontró el desafío muy estimulante; probó un poco al azar:

—¿Y si hacemos `n % 100` no obtendremos las decenas...? —dijo, mientras tomaba el teclado para sí.

```
echo (1971 % 100);
```

```

Consola
Compiling design (CSG Tree generation)...
ECHO: 71
Compiling design (CSG Products generation)

```

Figura 25.3: El resto de la división por 100 devuelve las decenas completas de un número.

Cecilia, ante el resultado de la figura 25.3, sintió que sus mejillas se encendían. Antonia no encontró reparos en reírse francaamente del desliz:

¹¡Hey! No me hagan *bullying*. (Nota del Editor)

²Disculpá; era una broma. (Nota de Antonia)

—Yo hice lo mismo —confesó—; hubiera sido muy hermoso que funcionara. Pero el resto de dividir por 100 te devuelve *todas* las decenas, no solamente el dígito de las mismas. Así que hay que buscar otra solución.

Cecilia se dispuso a esperar que Antonia le revelara una nueva función mágica, pero su amiga se anticipó:

—Puede resolverse usando la función ‘resto’ —y no dijo más.

Cecilia se sintió tocada en su amor propio; concentró su mirada en el monitor. «Muy bien» —pensó—, «si hago $n \% 10$ obtengo las unidades de n . Yo quiero el dígito que corresponde a las decenas. ¡Ay!, si solamente ese dígito ocupara el lugar de las unidades, podría rescatarlo como una unidad más...».

Cecilia frunció el ceño para obligarse a pensar, y cruzó sus brazos contra el pecho; su vista ya se perdía más allá del monitor de la computadora. De pronto y una vez más la epifanía ocurrió:

—¡Claro! —dijo en voz alta, sin darse cuenta—. ¡Puedo llevar las decenas al lugar de las unidades! ¡Simplemente debo dividir previamente el número por 10! —y sus dedos volaron sobre el teclado.

```
echo( (1971/10) %10 );
```

Consola
Compiling design (CSG Tree generation)...
ECHO: 7.1
Compiling design (CSG Product generation)

Figura 25.4: Primer intento (fallido) de Cecilia para rescatar el dígito de las decenas.

Un gesto de desazón cubrió el rostro de Cecilia mientras se desplomaba contra el respaldo de su silla.

—Corriste *todo* el número hacia la derecha, y eso incluía a las unidades, que ahora aparecen como un decimal —Antonia confirmó el diagnóstico que Cecilia podía ver por sí misma claramente—.

La idea está bien; sólo debemos deshacernos de ese molesto decimal.

25.2. FUNCIONES DE REDONDEO Y TRUNCAMIENTO

—Para aliviar a un número de decimales resultan muy útiles las funciones `round`, `floor` y `ceil` —explicó Antonia—. La primera redondea al número entero más cercano.

```
echo(round(3.14), round(2.72), round(-3.14),  
      round(-2.72));
```



Figura 25.5: Función `round`.

»`floor` te redondea ‘para abajo’.

```
echo(floor(3.14), floor(2.72), floor(-3.14),  
      floor(-2.72));
```



Figura 25.6: Función `floor`.

»Por su parte, `ceil` redondea ‘hacia arriba’.

```
echo(ceil(3.14), ceil(2.72), ceil(-3.14),  
      ceil(-2.72));
```

Cecilia sintió que recobraba el entusiasmo; recorrió las tres funciones para decidir con cuál aligerar su propia función de los

Figura 25.7: Función `ceil`.

indeseables decimales, pero tras unos instantes se sintió confundida: ninguna parecía cumplir con sus deseos. Necesitaba una función que siempre pudiera los decimales; algo que tomara los valores 3.14, 2.72, -3.14 y -2.72 y devolviera 3, 2, -3 y -2, respectivamente. Cuando giró, perpleja, en dirección a Antonia, la encontró sonriendo, como si hubiera adivinado sus pensamientos:

—Como ya sin duda descubriste, ninguna de esas funciones basta para resolver por sí sola nuestro problema. La función que necesitamos es una que trunque un número sin miramientos —confirmó Antonia—. Pero, curiosamente, esa función en OpenSCAD no existe..: ¡Mejor! —agregó, con una sonrisa exultante—. Así la creamos nosotras. En todos los lenguajes de programación que conozco esa función se denomina `truncate`; podemos usar ese mismo nombre, si te parece.

Cecilia no se sentía con ganas de discutirlo: sólo quería seguir con el módulo `hora_solar`, que tan lejos había quedado.

Antonia, ante el mutismo de Cecilia, le señaló con el mentón el teclado. Ésta, con un suspiro de resignación, lo tomó dispuesta a escribir una nueva función que supliera la curiosa omisión del lenguaje de OpenSCAD. Pero antes, se permitió una queja:

—Antonia, me dijiste hace unos momentos que sólo era necesaria la función resto, y ahora resulta que también hace falta una inexistente función de truncamiento...

—Soy docente —replicó Antonia, con gesto burlón—: ¿cuándo aceptarás que no podés creer en todo lo que digo?

Cecilia tampoco quiso discutir esto; volvió su mirada al monitor y enfocóse en el nuevo problema que tenía delante.

«Muy bien, Cecilia» —pensó—. «La función `floor` parece funcionar para números positivos, mientras que `ceil` es apropiada para los negativos. Si pudiera decidir, dentro de la función, qué tipo de redondeo aplicar dependiendo del signo del número recibido...» —no pasó mucho tiempo hasta que se le ocurrió probar con el operador ternario '`? :`'.

```
1 function truncate(n)=  
2   n<0 ? ceil(n) : floor(n);  
3  
4 echo(truncate(3.14), truncate(2.72),  
      truncate(-3.14), truncate(-2.72));
```



The screenshot shows a terminal window titled "Consola". The output of the command "echo" is displayed, showing the results of the truncate function for various inputs: 3, 2, -3, and -2. The output is "ECHO: 3, 2, -3, -2". Below the terminal window, there is some small, partially visible text that appears to be part of the terminal interface.

Figura 25.8: Cecilia crea la función `truncate`.

Cecilia estuvo a punto de saltar de su silla: el código no sólo funcionaba, sino también le parecía muy hermoso. La nueva función `truncate` se fijaba si el número recibido era menor que 0: en ese caso, devolvía `ceil(n)`; en caso contrario, `floor(n)`.

Antonia aplaudió lentamente, con gesto de feliz admiración:

—Excelente, Cecilia, pero no cantemos victoria: ¿funciona con 0? Siempre hay que probar los valores críticos...

Cecilia sintió que un nudo le apretaba la garganta; con un vago temor, hizo la prueba.

```
1 echo(truncate(0));
```



The screenshot shows a terminal window titled "Consola". The output of the command "echo" is displayed, showing the result of the truncate function for the input 0, which is 0. The output is "ECHO: 0". Below the terminal window, there is some small, partially visible text that appears to be part of the terminal interface.

Figura 25.9: Cecilia verifica cándidamente que `truncate(0)=0`.

Mientras Antonia reía alegre y francamente, Cecilia no pudo menos que sonreír al advertir la trampa inocente en la que había caído: cualquiera de las dos funciones hubiera servido con 0.

—Bueno, Cecilia; creo que ahora podemos obtener las decenas de cualquier número, ¿no? —preguntó retóricamente Antonia.

Cecilia estaba de acuerdo de todo corazón.

```
echo( truncate(1971/10) %10, truncate(-753/10) %10  
);
```

The screenshot shows a terminal window with the title 'Consola'. Inside, there is a message about 'Compiling design (CSG Tree generation)...', followed by 'ECHO: 7, -5', and another message about 'Compiling design (CSG Products generation)'.

Figura 25.10: Cecilia comprueba que puede obtener las decenas de cualquier número.

Antes de que interviniéra Antonia, Cecilia añadió:

—Y si queremos obtener las centenas, la manera es evidente.

```
echo( truncate(2021/100) %10,  
      truncate(-480/100) %10 );
```

The screenshot shows a terminal window with the title 'Consola'. Inside, there is a message about 'Compiling design (CSG Tree generation)...', followed by 'ECHO: 0, -4', and another message about 'Compiling design (CSG Products generation)'.

Figura 25.11: Obtención de las centenas.

—Perfecto —confirmó Antonia, tomando ahora el teclado—.
Pasemos en limpio, entonces:

Para las unidades: `truncate(n/1) %10`

Para las decenas: `truncate(n/10) %10`

Para las centenas: `truncate(n/100) %10`

Etc...

—¿Por qué dividiste n por 1 para las unidades? —preguntó Cecilia con visible extrañeza.

—Para hacer evidente el patrón que necesitamos aprehender a fin de escribir la función `n_a_dígito` de manera general —justificó Antonia.

Cecilia lo encontró razonable; supuso que el empleo de `truncate` también para las unidades tenía la misma explicación. Sin que Antonia se lo sugiriera, tomó el teclado mientras pensaba: «La función que voy a escribir recibirá dos parámetros: `n` y `p`. El primero es el número a analizar y el otro es el dígito que quiero rescatar, de manera que `p=0` indica las unidades, `p=1` las decenas, etc.» Con la mirada fija en la tabla escrita por Antonia, siguió sumida en sus pensamientos—. «La fórmula es la misma para todos las posiciones, salvo por el valor que divide a `n` antes del truncamiento: 1 cuando `p=0`, 10 cuando `p=1`, 100 para `p=2`... Son todas potencias de 10...», y la epifanía volvió a ocurrir, ya por tercera vez en el capítulo:

—¡Claro! ¡El divisor de `n` es igual a 10^p en todos los casos! —afirmó Cecilia ahora en voz alta, y sin esperar la confirmación de Antonia se lanzó a escribir la función.

```
1 // p es la posicion del dígito a obtener:  
2 // 0 para las unidades, 1 para las decenas, etc.  
3 function n_a_dígito(n,p)=  
4     truncate(n/pow(10,p))%10;  
5  
6 echo(n_a_dígito(1971,0), n_a_dígito(-1986,1),  
      n_a_dígito(2021,2));
```



Consola
Compiling design (CSG Tree generation)..
ECHO: 1, -8, 0
Compiling design (CSG Product generation)

Figura 25.12: Función que rescata un dígito de un número.

Cecilia, una vez más, estuvo a punto de gritar de alegría. Miró a Antonia con gesto radiante, y encontró en el rostro de su amiga

el reflejo de su propia felicidad. Sin embargo, tras unos instantes, Antonia sumó a su sonrisa una sombra de malicia:

—Cecilia, ¿no sería lindo que nuestra función fuera capaz de extraer, además, los dígitos que forman los decimales de un número? Quiero decir que, por ejemplo, haciendo `n_a_digito(3.1416, -1)` obtuviéramos 1.

Cecilia pensó que quizá loería, pero a esta altura no tenía tantas ganas de meter mano en su hermosa y flamante función. Además, no parecía que fuera necesaria esa funcionalidad extra para manipular las horas del reloj, que sólo serían valores enteros.

Con la sonrisa aún más marcada, Antonia preguntó con inocencia:

—¿Me dejás probar..?

Cecilia, de puro educada, le cedió sin entusiasmo y algo de aprensión el teclado. Pero Antonia no modificó en nada su función; tan sólo escribió:

```
echo(n_a_digito(3.1416, -1),  
      n_a_digito(-3.1416, -2));
```

The screenshot shows a terminal window with a light gray background. On the left, there is a vertical dark gray bar. The main area has a light gray header bar with the word "Consola" in black text. Below the header, the terminal output is displayed in a white area with black text. The output starts with "Compiling design (CSG Tree generation)...", followed by "ECHO: 1, -4", and then "Compiling design (CSG Product generation)".

Figura 25.13: La función `n_a_digito` es capaz de devolver también dígitos decimales.

Tras un instante de perplejo silencio, Cecilia cubrió sus ojos con una mano, mientras Antonia volvía a reír con ganas. «¡Por supuesto!» —pensó Cecilia— «Caí otra vez: el código ya abarca todos los dígitos posibles, puesto que dividir por 10^{-2} , por ejemplo, es lo mismo que multiplicar por 10^2 , lo cual no hace más que llevar el segundo decimal al lugar ‘mágico’ de las unidades».

Cuando Antonia dejó de reír encontró a Cecilia con ganas de continuar.

—Bien —dijo ésta, recuperando el teclado—; creo que ya podemos empezar a completar nuestro ya casi olvidado módulo hora_solar, ¿no te parece?

```
1 module hora_solar(horas,minutos){  
2     alfa=alfa(horas+minutos/60);  
3     hora_decenas=n_a_dígito(horas,1);  
4     hora_unidades=n_a_dígito(horas,0);  
5     minuto_decenas=n_a_dígito(minutos,1);  
6     minuto_unidades=n_a_dígito(minutos,0);  
7  
8     echo(hora_decenas, hora_unidades,  
         minuto_decenas, minuto_unidades);  
9 }
```

»La línea 8 la añadí sólo para comprobar que todo funciona como es debido —aclaró Cecilia—. ¿A ver si es así..?

```
hora_solar(12,30);
```



Figura 25.14: Cecilia comprueba que la función hora_solar consigue los dígitos de la hora y los minutos.

Parecía que sí.

Una hora escrita con Sol – II

AL DÍA SIGUIENTE, Antonia esperaba a Cecilia con impaciencia: —¡Por fin llegaste! —saludó, fingiendo burlona intranquilidad.

—No te preocupes —respondió su amiga con una sonrisa—; en cuanto imprimamos nuestro reloj podrás medir con precisión mi impuntualidad... al menos cuando no esté nublado.

Cecilia se sentó frente a la computadora con menos entusiasmo que de costumbre; quizás ella también, ante la inminente finalización de esta aventura, estaba sintiendo una nostalgia anticipada.

Antonia, según su costumbre, pareció no notarlo:

—Ya sabemos cómo calcular el ángulo con el que debemos escorzar nuestros dígitos a cada hora puntual; sabemos también cómo rescatarlos individualmente. Ahora parecería que sólo debemos ponerlos uno al lado del otro, y recorrer las horas y minutos desde las 6:00 a las 18:00.

Antonia puso un énfasis involuntario en el último potencial, que provocó en Cecilia una instintiva actitud defensiva: anticipó que otro problema estaba por hacer su aparición.

26.1. EL SEPARADOR

»En primer lugar, tenemos entonces que colocar los cuatro dígitos que componen la hora uno al lado del otro —continuó Antonia.

—¡No te olvides del separador de horas y minutos! —interrumpió Cecilia que estaba alerta—. Las 1345 no parecen una hora muy seria...

—Tenés razón —admitió Antonia, sonriendo—. Tenemos que hacerle lugar también. No sólo en el reloj, sino en el texto: se merece su propio módulo —e inmediatamente deslizó sus dedos sobre el teclado.

```
1 module separador(alfa){  
2     for(i=[-1,1])  
3         translate([i*0.5*(alto_pixel+delta_alto), 0,  
4                     -.01])  
5         rayo_de_sol(alfa);  
6     }  
7     difference(){  
8         rotate([-90,0,0])  
9         semicilindro(150,30,center=true);  
10        separador(60);  
11        translate([0,-30,0])  
12        digito(60,2);  
13    }
```

—¿Creo que no tiene mucho misterio, no? —preguntó Antonia retóricamente—. El separador consiste en dos rayos de Sol, dispuestos vertical y simétricamente.

—Es cierto —admitió Cecilia—; pero recordemos que nuestro separador convivirá con la `hora_solar`, que recibirá como parámetros la hora y los minutos, no el ángulo —agregó, mientras se disponía a corregir el módulo de Antonia.

```
1 module separador(horas,minutos){  
2     alfa=alfa(horas+minutos/60);
```

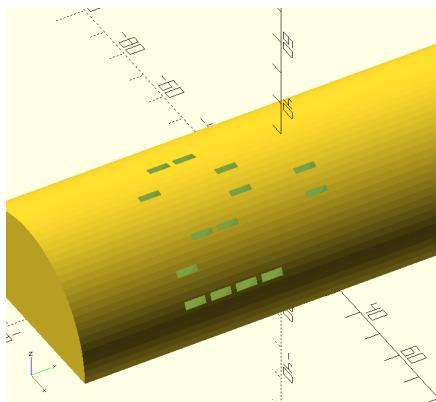


Figura 26.1: Primera versión del separador de horas y minutos.

```

3   for(i=[-1,1])
4     translate([i*0.5*(alto_pixel+delta_alto), 0,
5       -.01])
6     rayo_de_sol(alfa);
7   }
8   difference(){
9     rotate([-90,0,0])
10    semicilindro(150,30,center=true);
11    separador(14,0);
12    translate([0,-30,0])
13    digito(60,2);

```

»A las 14:00 les corresponden un ángulo de 60° en nuestro hemisferio; por eso el separador quedó alineado con el '2' —comentó Cecilia, mientras Antonia la miraba en silencio y como si no la reconociese.

26.2. EL CUERPO DEL RELOJ

Cecilia seguía mirando el texto con gesto de insatisfacción:

—¿No te parece que el cuerpo del reloj ya está mereciendo también su propio módulo? Las crudas líneas 8 y 9 me parecen indignas de convivir con el separador y el digito... —y tras unos instantes, compartió con Antonia su nuevo módulo.

```
1 module cuerpo(largo){  
2     rotate([-90,0,0])  
3         semicilindro(largo,radio_semicilindro,  
4                         center=true);  
5     }  
6     difference(){  
7         cuerpo(150);  
8         separador(14,0);  
9         translate([0,-30,0])  
10        digito(60,2);  
11    }
```

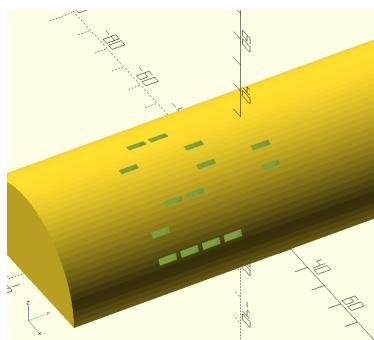


Figura 26.2: Cecilia otorga al cuerpo del reloj su merecido módulo propio.

26.3. LA POSICIÓN DE LOS DÍGITOS

LAS HORAS

Antonia recuperó el teclado casi con sigilo:

—Muy bien, Cecilia —deslizó con suavidad—. Ahora sí, creo que podemos tratar de colocar los dígitos; comenzemos por la hora —añadió, mientras elaboraba el sencillo boceto de la figura 26.3.

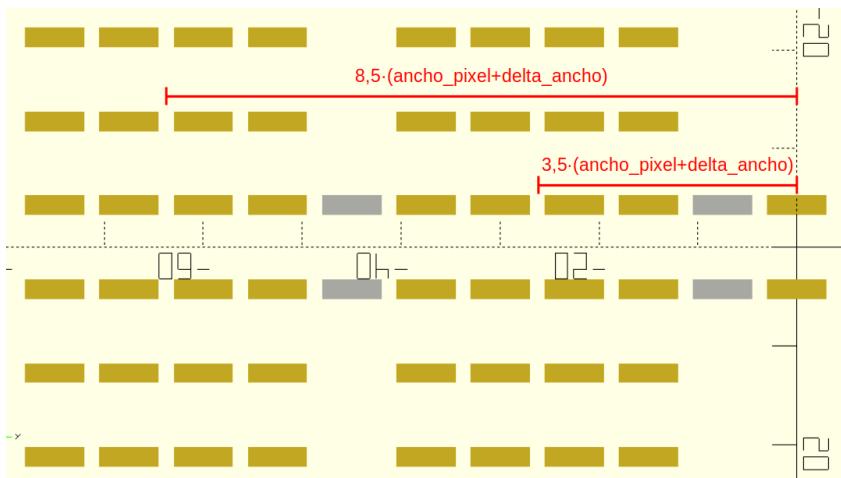


Figura 26.3: Antonia calcula la posición de los dígitos de las horas.

»Agregué dos separadores ‘fantasma’ para ayudarnos a calcular la ubicación de los centros de ambos dígitos; si no me equivoco, las decenas de las horas se ubican a $8.5 * (\text{ancho_pixel} + \text{delta_ancho})$ del origen de coordenadas, mientras que las unidades están a $3.5 * (\text{ancho_pixel} + \text{delta_ancho})$ del mismo punto.

Cecilia recorrió con la punta de un dedo el monitor mientras contaba en silencio: enseguida pudo confirmar los resultados de Antonia.

—Si te parece bien, no demoremos más en agregar las horas a nuestro reloj... —con una amplia sonrisa, Antonia sumó al módulo `hora_solar` la nueva conquista:

```
1 module hora_solar(horas, minutos){
```

```
2 alfa=alfa(horas+minutos/60);
3 hora_decenas=n_a_digito(horas,1);
4 hora_unidades=n_a_digito(horas,0);
5 minuto_decenas=n_a_digito(minutos,1);
6 minuto_unidades=n_a_digito(minutos,0);
7
8 delta_y=ancho_pixel+delta_ancho;
9 translate([0,-8.5*delta_y,0])
10 digito(alfa,hora_decenas);
11 translate([0,-3.5*delta_y,0])
12 digito(alfa,hora_unidades);
13 }
14 difference(){
15 cuerpo(170);
16 separador(12,0);
17 hora_solar(12,0);
18 }
```

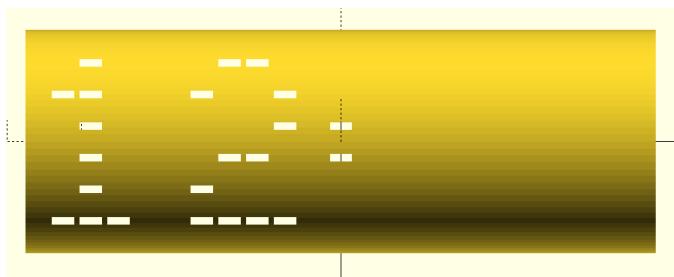


Figura 26.4: Antonia agrega las horas al módulo hora_solar.

Cecilia tuvo que reprimir un grito de alegría ante la figura 26.4: aún no se acostumbraba a que la realidad —aun tratándose de una realidad “computacional”— se adecuara fielmente a las ideas y cálculos que elucubraban con Antonia. Se preguntó, no sin inquietud, qué sentiría cuando vieran al fin el reloj impreso, como un objeto real y plenamente físico... Tomando para sí el teclado, decidió probar el módulo con otras horas: las 9:00 y las 15:00.

```
1 difference(){
2     cuerpo(170);
3     separador(9,0);
4     hora_solar(9,0);
5 }
```

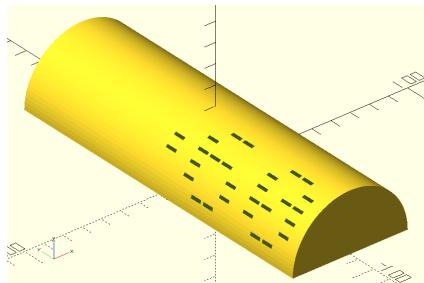


Figura 26.5: Cecilia comprueba el módulo hora_solar para las 9:00.

```
1 difference(){
2     cuerpo(170);
3     separador(15,0);
4     hora_solar(15,0);
5 }
```

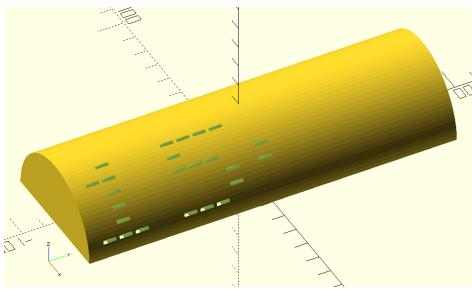


Figura 26.6: Cecilia comprueba el módulo hora_solar para las 15:00.

Las 9:00 y las 15:00 inquietaron un poco a Cecilia: ¿No estaban demasiado cerca de la base del reloj? Temió que más cerca de las 18:00 o de las 6:00 los dígitos se “cayeran” del mismo... Luego de unos instantes de aprensión, se armó de coraje y decidió probar una hora extrema:

```
1 difference() {
2     cuerpo(170);
3     separador(17,0);
4     hora_solar(17,0);
5 }
```

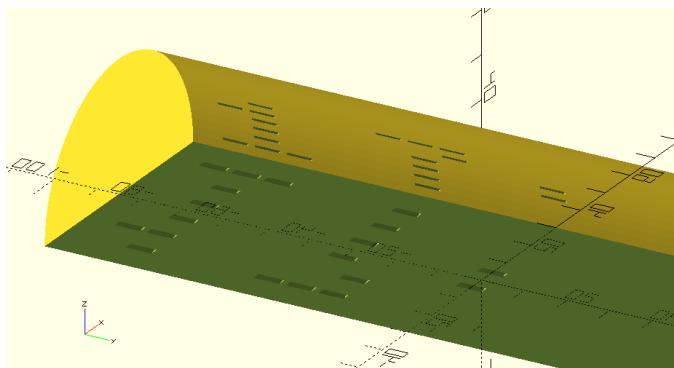


Figura 26.7: Cecilia temía infundadamente que hora_solar no superara la prueba de las 17:00.

Cecilia sintió, una vez más y ahora ante la figura 26.7, que sus mejillas se encendían de rubor: ¿cómo fue capaz de desconfiar de sus fórmulas y ecuaciones? ¡No podían fallar! Los rayos respondían fielmente y con seguridad, sin importar el ángulo solicitado. Giró para mirar a Antonia, a quien encontró con una sonrisa en la que flotaba una tierna comprensión: seguramente ella también había pasado (o quizás seguía sintiendo, alguna que otra vez) esas breves aunque inquietantes desconfianzas.

LOS MINUTOS

—¡Es hora de ir a por los minutos! —propuso Cecilia, agitando ligeramente la cabeza como si quisiera despejarla de sus últimas inquietudes—. Están ubicados simétricamente con respecto a las horas; yo diría que la cosa ya está prácticamente resuelta:

```
1 module hora_solar(horas,minutos){  
2     alfa=alfa(horas+minutos/60);  
3     hora_decenas=n_a_digito(horas,1);  
4     hora_unidades=n_a_digito(horas,0);  
5     minuto_decenas=n_a_digito(minutos,1);  
6     minuto_unidades=n_a_digito(minutos,0);  
7     delta_y=ancho_pixel+delta_ancho;  
8     // horas  
9     translate([0,-8.5*delta_y,0])  
10    digito(alfa,hora_decenas);  
11    translate([0,-3.5*delta_y,0])  
12    digito(alfa,hora_unidades);  
13    // minutos  
14    translate([0,3.5*delta_y,0])  
15    digito(alfa,minuto_decenas);  
16    translate([0,8.5*delta_y,0])  
17    digito(alfa,minuto_unidades);  
18 }  
19 difference(){  
20     cuerpo(170);  
21     separador(12,34);  
22     hora_solar(12,34);  
23 }
```

Contemplando embelesada la figura 26.8, Cecilia sintió que la programación no podría hacerla nunca más dichosa que en ese momento.

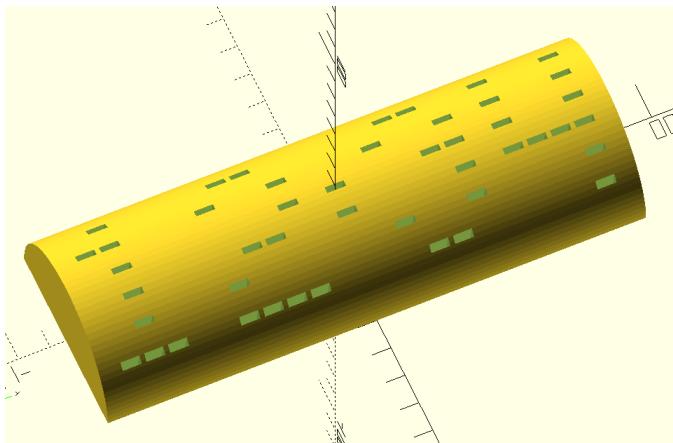


Figura 26.8: Cecilia estampa las 12:34.

UN CERO INICIAL

—¿No te parece un poco artificial la aparición de un cero en las horas previas a las 10:00? —preguntó Antonia, con una clara entonación retórica, mientras recobraba el teclado.

```
1 difference() {
2     cuerpo(170);
3     separador(8,53);
4     hora_solar(8,53);
5 }
```

—¿Sugerís que suprimamos la aparición del mismo? —preguntó a su vez Cecilia, dando a entender que le parecía una buena idea, y tomando el teclado.

```
1 module hora_solar(horas,minutos){
2     alfa=alfa(horas+minutos/60);
3     hora_decenas=n_a_digito(horas,1);
4     hora_unidades=n_a_digito(horas,0);
5     minuto_decenas=n_a_digito(minutos,1);
6     minuto_unidades=n_a_digito(minutos,0);
```

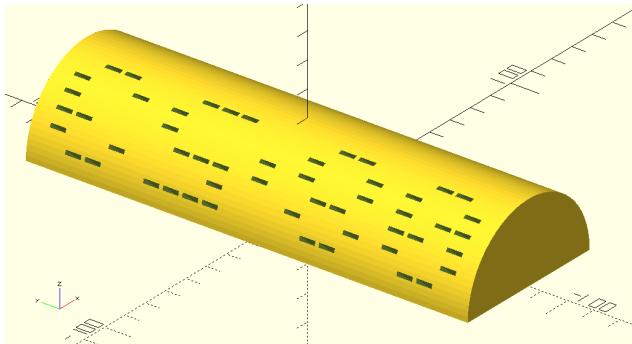


Figura 26.9: Antonia opina que las 8:53 no deberían ostentar un cero inicial.

```

7   delta_y=ancho_pixel+delta_ancho;
8   // horas
9   if (hora_decenas != 0)
10     translate([0,-8.5*delta_y,0])
11     digito(alfa,hora_decenas);
12   translate([0,-3.5*delta_y,0])
13     digito(alfa,hora_unidades);
14   // minutos
15   translate([0,3.5*delta_y,0])
16     digito(alfa,minuto_decenas);
17   translate([0,8.5*delta_y,0])
18     digito(alfa,minuto_unidades);
19 }
20 difference(){
21   cuerpo(170);
22   separador(8,53);
23   hora_solar(8,53);
24 }
```

Cecilia contempló su último retoque del texto con indisimulada satisfacción: en la línea 9 indagaba si las decenas de la hora eran distintas de cero; sólo en ese caso se horadaba el cuerpo del

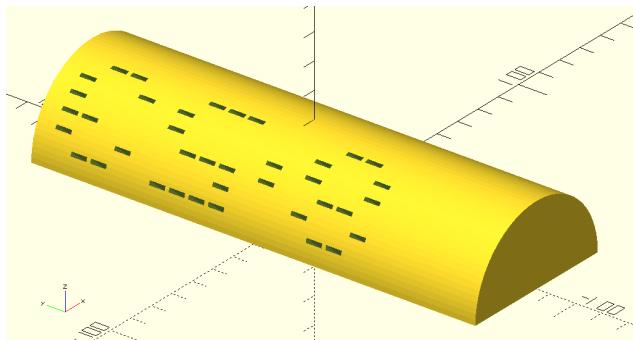


Figura 26.10: Cecilia omite el ‘0’ en las horas previas a las 10:00.

reloj con el correspondiente dígito. Por las dudas lo probó con una hora posterior a las 10:00:

```
1 difference () {  
2     cuerpo (170) ;  
3     separador (14 ,39) ;  
4     hora_solar (14 ,39) ;  
5 }
```

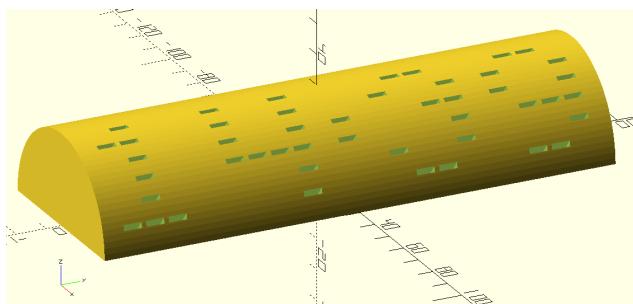


Figura 26.11: Cecilia, prudentemente, confirma que las decenas de las horas se siguen mostrando para las horas posteriores a las 10:00.

Todo seguía en orden.

26.4. EL SEPARADOR – II

Antonia y Cecilia contemplaban en silencio su obra; cualquiera que las viese no hubiera podido distinguir si estaban serenamente felices por el claro resultado conquistado o íntimamente inquietas en busca de algún detalle para mejorar o agregar. Quizá se debatieran entre ambos sentimientos.

—Creo que el separador es parte de la hora_solar; debería pertenecer a este módulo —soltó Antonia suavemente.

Cecilia asintió en silencio, mientras se disponía a escribir:

```

1 module hora_solar(horas,minutos){
2     alfa=alfa(horas+minutos/60);
3     hora_decenas=n_a_dígito(horas,1);
4     hora_unidades=n_a_dígito(horas,0);
5     minuto_decenas=n_a_dígito(minutos,1);
6     minuto_unidades=n_a_dígito(minutos,0);
7     delta_y=ancho_pixel+delta_ancho;
8     // horas
9     if (hora_decenas != 0)
10        translate([0,-8.5*delta_y,0])
11        dígito(alfa,hora_decenas);
12        translate([0,-3.5*delta_y,0])
13        dígito(alfa,hora_unidades);
14     // minutos
15        translate([0,3.5*delta_y,0])
16        dígito(alfa,minuto_decenas);
17        translate([0,8.5*delta_y,0])
18        dígito(alfa,minuto_unidades);
19     // separador
20     separador(horas,minutos);
21 }
22 difference(){
23     cuerpo(170);
24     hora_solar(13,46);
25 }
```

Cecilia y Antonia se sentían cada vez más felices. Mientras tanto, un nuevo *bug* que las acechaba desde el capítulo 18 las esperaba con impaciencia en el próximo.

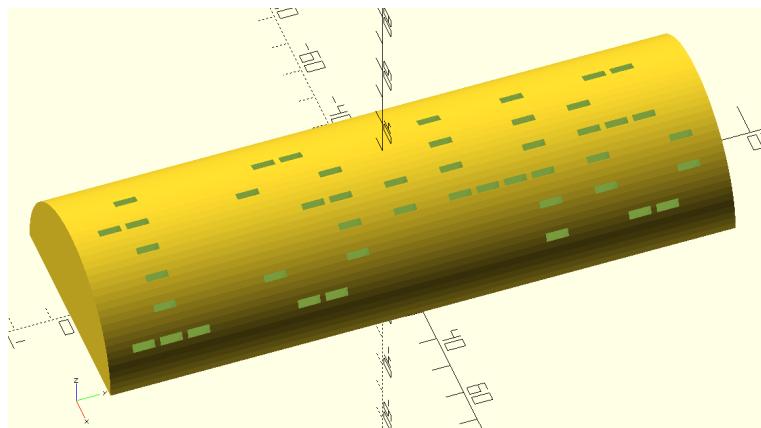


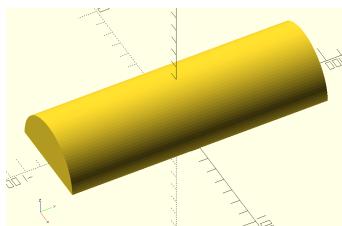
Figura 26.12: Cecilia y Antonia comprueban una vez más el funcionamiento del módulo hora_solar, ajenas por el momento a la inminente manifestación, en el capítulo siguiente, de un oscuro *bug* escondido en su texto.

El reloj de Sol – I

EL DÍA SIGUIENTE encontró a Cecilia y Antonia ansiosas por atravesar el cuerpo del reloj con todas las horas diurnas. Cecilia incluso podía sentir en sus manos una delicada inquietud, casi un fervor, que las sostenía encima del teclado con una suerte de tensión, como si estuvieran al acecho. Con voz vibrante anunció a su amiga:

—Creo que ya es tiempo de levantar, con todos los ladrillos que cocimos hasta aquí, el edificio por el que suspiramos desde hace 26 capítulos.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(170);
4         // HACER: Las horas.
5     }
6 }
7 reloj_de_sol();
```



—¿Por qué ‘170’? —objetó Antonia—. Si bien es cierto que en el capítulo anterior ese valor particular nos sirvió, pensemos que un eventual lector de nuestro texto quizás quiera modificar los valores de `ancho_pixel` o `delta_ancho`, lo cual tornará obsoleto el

caprichosamente específico valor 170.

Cecilia expresó su acuerdo frunciendo los labios; rápidamente esbozó un dibujo que las ayudara a establecer el largo del cuerpo del reloj de manera general.

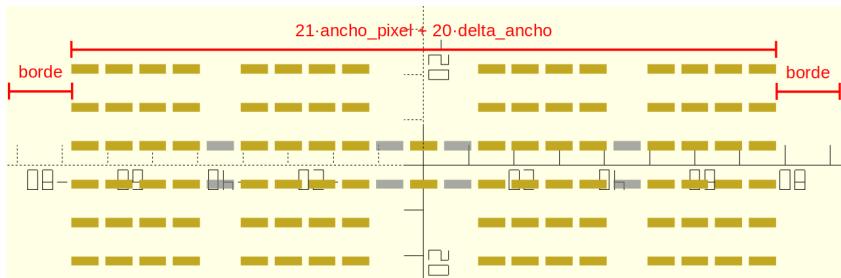


Figura 27.1: Largo del cuerpo del reloj.

—Si no me equivoco, los dígitos con los separadores ocupan $21*\text{ancho_pixel}+20*\text{delta_ancho}$ —aventuró Cecilia.

Esta vez fue Antonia quien recorrió en silencio y con la punta de sus dedos el monitor, mientras confirmaba la cuenta de Cecilia:

—Está bueno, además, que permitamos al lector decidir el tamaño del borde. Quizá quiera escribir en él alguna leyenda: después de todo, hay una larga tradición de acompañar los relojes de Sol con frases más o menos profundas... o al menos ingeniosas —adelantó Antonia.

Cecilia abrió grandes los ojos, dirigiéndolos a su amiga:

—¿Se puede construir texto en OpenSCAD? —y mientras lo preguntaba, se dio cuenta de que no podía ser de otra manera: ¿Cómo podían dejar de lado esa posibilidad los creadores de este hermoso, potente y dúctil lenguaje..?

—Claro —confirmó Antonia con una sonrisa—; lo veremos en el penúltimo capítulo: faltan apenas uno o dos. Bah, en realidad será en el último: luego vendrá apenas un epílogo, lleno exclusivamente de pésima literatura y con una sorpresa.

Cecilia recordó la incomodidad que sintiera, en los primeros capítulos, acerca de las referencias a sus reuniones como secciones de un manual. Curiosamente, y sin saber muy bien porqué, esas alusiones ya no la inquietaban tanto.

Antonia, por su parte, ya estaba escribiendo.

```
1 borde = 0;
2 largo_reloj = 21*ancho_pixel + 20*delta_ancho +
   2*borde;
3 module reloj_de_sol(){
4   difference(){
5     cuerpo(largo_reloj);
6     hora_solar(12,0);
7   }
8 }
9 reloj_de_sol();
```

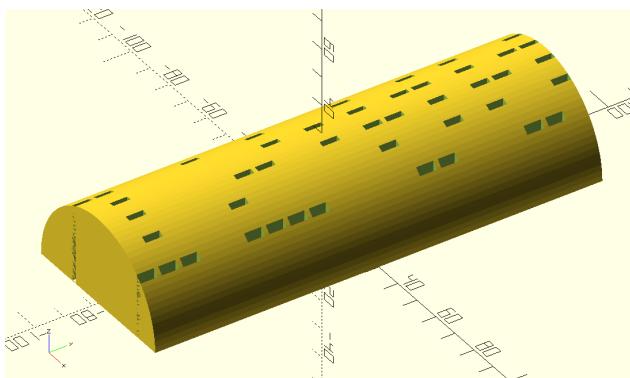


Figura 27.2: Antonia confirma el largo del cuerpo del reloj.

—Ahí podemos ver que el '1' está arañando la pared izquierda del reloj —señaló Antonia en la figura 27.2—; es lógico, ya que usé un borde nulo a propósito. Lo importante es que las medidas evidentemente funcionan. Moveré las variables al principio, junto

a las demás, al tiempo que le otorgo a borde un valor más sensato —anunció Antonia.

```
1 hemisferio="sur";
2 alto_pixel = 2;
3 ancho_pixel = 6;
4 delta_alto = 6.5;
5 delta_ancho = 1.5;
6 borde = 10;
7 radio_semicilindro = 30;
8 H = radio_semicilindro+10;
9 largo_reloj = 21*ancho_pixel + 20*delta_ancho +
   2*borde;
10
11 //    Mucho texto en el medio
12
13 module reloj_de_sol(){
14     difference(){
15         cuerpo(largo_reloj);
16         hora_solar(12,0);
17     }
18 }
19 reloj_de_sol();
```

27.1. UN VIEJO *bug* AGAZAPADO HACE FINALMENTE SU APARICIÓN

Cecilia se encontraba exultante; tomó el teclado casi con vehemencia, decidida a agujerear el reloj con todas las horas desde las 6:00 hasta las 18:00.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         for(horas=[6:17],
5             minutos=[0:59])
```

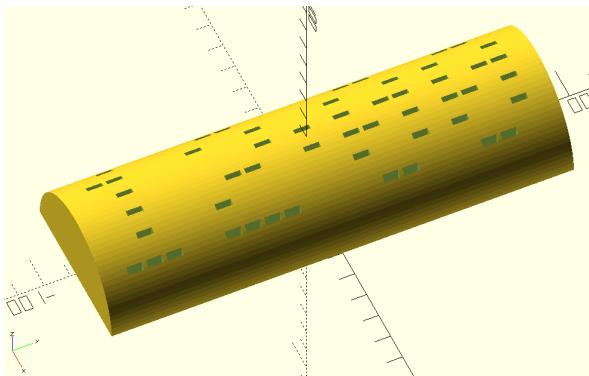


Figura 27.3: El largo del reloj abarca ahora sendos bordes.

```
6             hora_solar(horas ,minutos ) ;
7     hora_solar(18 ,0 ) ;
8 }
9 }
10 reloj_de_sol();
```

Quizá para crear un clima de tensión dramática, o quizás para satisfacer la inconsciente y paradójica necesidad que todos, alguna vez, sentimos por retardar la finalización de un proyecto largamente acariciado, Cecilia contempló largamente su texto antes de pulsar la definitiva tecla . Las flamantes líneas 4 y 5 creaban dos variables, horas y minutos, las cuales recorrían los valores enteros desde el 6 al 17 y desde el 0 hasta el 59, respectivamente. Para cada combinación de ambas variables (en otras palabras, para todas las horas posibles entre las 6:00 y las 17:59) se restaba al cuerpo del reloj una hora_solar. Finalmente se restaba la hora final: las 18:00.

Con la yema del dedo índice ligera y nerviosamente apoyada sobre la tecla , Cecilia dirigió la mirada a su amiga, en cuyos ojos creyó percibir el brillo de la misma emoción. En perfecto

silencio, y reclinando levemente la espalda contra el respaldo de la silla, presionó dicha tecla, dispuesta plenamente a recibir en sus ojos la felicidad y la dicha bajo la forma de un hermoso reloj de Sol digital, listo para ser impreso.

Sin embargo, tras unos cuantos minutos fue una desagradable e inesperada sorpresa la que se desplegó en el monitor frente a su desconcertada mirada. La figura 27.4, por un lado, mostraba el reloj casi uniformemente barrido por una impiadosa cuchilla, que eliminó del mismo limpias rebanadas paralelas.

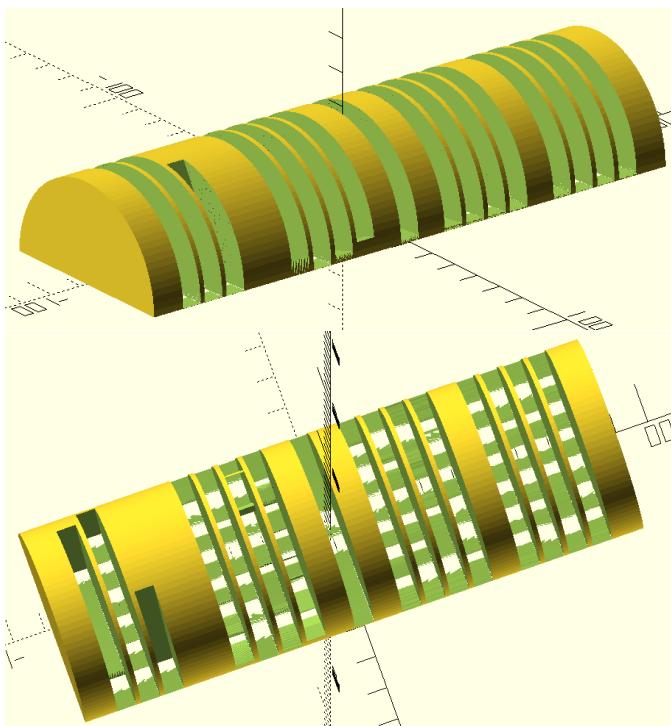


Figura 27.4: Un viejo *bug* hace finalmente su aparición.

El otro problema le pareció incluso más urgente y preocupante:

en la consola de mensajes un error campeaba, en un alarmante color rojo y por duplicado, tal como se puede apreciar en la figura 27.5.

```
Consola
Saved backup file: /home/luis/snap/openscad/182/local/share/OpenSCAD/backups/reloj-de-sol-digital-backup
Compiling design (CSG Tree generation)...
Compiling design (CSG Products generation)...
ERROR: Unable to convert points[1] = [inf, 40] to a vec2 of numbers in file reloj-de-sol-digital.scad, line 32
ERROR: Unable to convert points[1] = [-inf, 40] to a vec2 of numbers in file reloj-de-sol-digital.scad, line 32
Geometries in cache: 6
Geometry cache size in bytes: 15376
CSG1 RelojDeSolDigital in cache: 0
```

Figura 27.5: Un ominoso mensaje de error aparece en la consola.

En los ojos de Cecilia el brillo confiado del entusiasmo se trocó en un opaco fulgor de inquieta desazón; giró en dirección a Antonia, en un instintivo y mudo pedido de auxilio. La encontró sonriendo: era una sonrisa franca, no burlona; en ella flotaba una delicada expresión de ternura. Cecilia no tardó en comprender que había caído ingenuamente en otra trampa, aunque no supo si se la había tendido el reloj o su amiga.

—Atendamos a un problema a la vez —empezó Antonia—; de acuerdo al mensaje de error de la consola, parece que hubo un problema en la línea 32 del texto —y con el mentón señaló el monitor. Cecilia entendió que eso era una invitación a volver sobre esa línea escrita tanto tiempo atrás.

```
28 //      translate([-w,ancho_pixel/2,0])
29 //      Ojo : borrar 'center=true' abajo
30 rotate ([90,0,0])
31     linear_extrude(ancho_pixel,center=true)
32         polygon(vertices);
33 }
34
35 ordinitos = r
```

Figura 27.6: De acuerdo al mensaje de la consola en la figura 27.5, la línea 32 es la presunta culpable del *bug*.

Cecilia no podía ver qué problema era capaz de ocurrir en línea

tan inocente como `polygon(vertices)`; su gesto de perplejidad fue tan evidente que hasta Antonia pudo apreciarlo.

—Leamos con atención el mensaje de error de la consola —propuso esta última con suavidad—. Parece que hay dos puntos (`points[1] = [inf, 40]` y `points[1] = [-inf, 40]`) que no pueden ser convertidos a un vector de números...

Antonia adoptaba en su descripción un tono evidentemente didáctico, en el que Cecilia leía la invitación a descifrar por sí misma el enigma que tenía delante, a la vez que entendía que el final del reloj se había alejado una vez más, y que la mejor actitud que debía adoptar era ni más ni menos que la que las llevó hasta aquí: las ganas de aprender y superar los obstáculos con confianza y decisión.

Tras volver a leer el mensaje de error, una pregunta se abrió paso con claridad en su mente:

—¿Qué representan `inf` y `-inf`? —inquirió, sospechando la respuesta.

—Infinito y menos infinito —respondió lacónicamente Antonia con una amplia sonrisa, sin duda siguiendo y aprobando el proceso mental de Cecilia.

—¿Y de dónde pudo haber surgido un infinito..? —se preguntó ésta en voz alta. Decidió que quizás sería buena idea indagar de dónde surgían los `vertices` que recibía `polygon`.

Los `vertices` procedían —como podía apreciarse en el texto en la figura 27.7— de las líneas 22 a 25; nada anómalo parecía haber en ellas: `alto_pixel` y `H` eran parámetros iniciales que no podían dar lugar a ninguna clase de infinito, y `D` era calculado en la línea 21... En ese momento, los ojos de Cecilia se abrieron como dos soles que quisieran abrasar el monitor: ¡En el cálculo de `D` se empleaba la función tangente! Si `tan(alfa)` fuera nulo...

—Antonia, ¿es posible que en OpenSCAD la división por cero arroje como resultado infinito? —conjeturó instintivamente Cecilia, respondiendo a una vehemente coronada.

```

17 radio_semicilindro = 50;
18 H = radio_semicilindro+10;
19
20 module rayo_de_sol(alfa){
21   D=H/tan(alfa);
22   vertices=[[-alto_pixel/2,0],
23             [D-alto_pixel/2,H],
24             [D+alto_pixel/2,H],
25             [alto_pixel/2,0]];
26   // TODO: medir la duracion de esta solucion
27   //       y la de esta otra:
28   //       translate([0,ancho_pixel/2,0])
29   //       Ojo : borrar 'center=true' abajo
30   rotate ([90,0,0])
31   linear_extrude(ancho_pixel,center=true)
32   polygon(vertices);
33 }
34
35 digits = 5

```

Figura 27.7: En busca del origen del bug.

La sonrisa de Antonia no podía ser más amplia:
—Sí, es exactamente así; mirá: —y tomando el teclado lo confirmó con un par de ejemplos.^{1,2,3,4,5}

Consola	
1 echo (1/0, -8/0);	Compiling design (CSG Tree generation)... ECHO: inf, -inf Compiling design (CSG Products generatio

Cecilia recobró gran parte de su confianza al haber dado al menos con el origen del error; ahora tenía por delante, eso sí, la tarea de solucionarlo. Recordó que `tan(alfa)` sólo podía valer 0 si `alfa==0` o `alfa==180`: ¿En qué condiciones los rayos de Sol caían

¹¡Ah, no! $\frac{1}{0} = \infty$! Esto es demasiado! ¡Renuncio! (Nota del Editor)

²No, editor... daaleee... quedate. (Nota de Cecilia, Antonia y Luis)

³Uh... ¿En serio quieren que me quede? (Nota del Editor)

⁴¡Pero claro que sí! (Nota de Antonia, Cecilia y Luis)

⁵Bueno: me quedo ☺. (Nota del Editor)

con esos ángulos? Buscó la respuesta en el capítulo 23; la encontró en la figura que copiamos como 27.8.

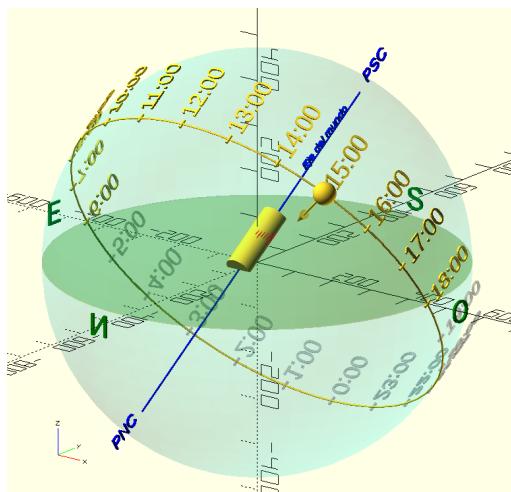


Figura 27.8: A las 6:00 y a las 18:00 los rayos del Sol forman un ángulo nulo respecto del plano XY.

—¡Claro! —Cecilia ya estaba nuevamente en vena—. Un ángulo de 0° o 180° corresponden a rayos rasantes respecto de la base del reloj... ¡O sea que ocurren a las 6:00 y a las 18:00!

Con una amplia sonrisa de satisfacción, erguiendo los hombros con renovada seguridad, Cecilia propuso:

—¡Listo, Antonia! Con evitar las 6:00 y las 18:00 solucionamos ese problema.

—Sí... —Antonia asintió con desgano—. Podemos hacer eso y desentendernos del mismo; pero el problema, considerado en sí mismo, seguirá ahí, en el texto.

—¿Pretendés acaso modificar el comportamiento de la función tangente? —preguntó Cecilia, extrañada y un poco mortificada por no haber recibido la felicitación que esperaba de su amiga.

—¡Claro que no! —protestó Antonia—. No me refiero a eso. Quiero decir que debemos anticipar que el eventual lector de nuestro texto no caiga en el mismo error y la misma perplejidad que nos detuvo por unos momentos. Y recordá que nosotras mismas seremos también nuevas lectoras, cuando nos enfrentemos a nuestra propia obra en un par de meses.

—Nadie se baña dos veces en el mismo río, ¿no? —comentó Cecilia, que sabía que esta frase de Heráclito era una de las preferidas de Antonia: al menos, una de las tantas que la acompañaban y le gustaba recordar.

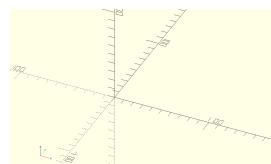
Antonia sonrió dulcemente, mientras asentía:

—Porque el río no es el mismo, ni una es la misma...

27.2. ASSERT

—En general, los mensajes de error de OpenSCAD, como ya pudiste comprobar, no sólo son bastante crípticos sino que rara vez apuntan al origen de los problemas: más bien señalan la línea donde finalmente se manifiestan y explotan —Antonia inició otra de sus explicaciones retomando el tono didáctico—. Ahora bien, cuando una sabe que cierta función o módulo debe recibir parámetros que respondan a determinadas condiciones, y que en caso de que no se cumplan el proceso de construcción no debe continuar, resulta utilísima la sentencia `assert` —y tecleó con rapidez otro de sus ejemplos vanales.

```
1 module prueba(a){
2     assert(a!=0,"el parametro
3         'a' no debe ser cero");
4     cube(20);
5 }
prueba(a=0);
```



Antonia permitió que Cecilia contemplara el texto y el mensaje

```
Consola
Compiling design (CSG Tree generation)...
ERROR: Assertion '(a != 0)' failed: "el parámetro 'a' no debe ser cero" in file aseveraciones.scad, line 2
TRACE: called by 'assert' in file aseveraciones.scad, line 2
TRACE: called by 'prueba' in file aseveraciones.scad, line 6
Compiling design (CSG Products generation)...
Geometries in cache: 7
Geometries cache size in bytes: 16104
```

Figura 27.9: Assert.

de la consola en la figura 27.9 unos momentos antes de continuar:

—La idea es ésta: `assert` admite dos parámetros: el primero es una condición a evaluar (en nuestro caso, `a!=0`). Si la misma es verdadera, todo bien: OpenSCAD prosigue con el resto de las instrucciones. Pero si es falsa, detiene la construcción con un mensaje de error, cuyo texto se corresponde con el segundo parámetro que vos le pasaste a `assert`. De esa manera, no sólo impedís que se lleve a cabo una construcción probablemente anómala sino que recibís un mensaje de error un poco más expresivo y certero.

A Cecilia le pareció bastante útil:

—Entonces se trata de anticipar los valores extremos `alfa==0` y `alfa==180` —dijo, y recobrando el teclado propuso la siguiente modificación:

```
1  module rayo_de_sol(alfa){
2      assert(alfa!=0 && alfa!=180, "'alfa' no debe
3          ser nulo ni llano.");
4      D=H/tan(alfa);
5      vertices=[[-alto_pixel/2,0],
6                  [D-alto_pixel/2,H],
7                  [D+alto_pixel/2,H],
8                  [alto_pixel/2,0]];
9      // TODO: medir la duracion de esta solucion
10     //       y la de esta otra:
11     //       translate([0,ancho_pixel/2,0])
12     //       Ojo : borrar 'center=true' abajo
13     rotate ([90,0,0])
14         linear_extrude(ancho_pixel,center=true)
```

```

14         polygon(vertices);
15     }
16 // Varias lineas de texto
17 module reloj_de_sol(){
18     difference(){
19         cuerpo(largo_reloj);
20         hora_solar(6,0);
21     }
22 }
23 reloj_de_sol();

```

```

Compiling design (CSG Tree generation)...
ERROR: Assertion `((alfa != 0) && (alfa != 180))' failed: "alfa' no debe ser nulo si llano." in file reloj-de-sol-digital.scad, line 21
TRACE: called by 'assert' in file reloj-de-sol-digital.scad, line 21
TRACE: called by 'rayo_de_sol' in file reloj-de-sol-digital.scad, line 105
TRACE: called by 'translate' in file reloj-de-sol-digital.scad, line 104
TRACE: called by 'if' in file reloj-de-sol-digital.scad, line 101
TRACE: called by 'for' in file reloj-de-sol-digital.scad, line 99
TRACE: called by 'digito' in file reloj-de-sol-digital.scad, line 136
TRACE: called by 'translate' in file reloj-de-sol-digital.scad, line 135
TRACE: called by 'if' in file reloj-de-sol-digital.scad, line 134
TRACE: called by 'hora_solar' in file reloj-de-sol-digital.scad, line 166
TRACE: called by 'difference' in file reloj-de-sol-digital.scad, line 161
TRACE: called by 'reloj_de_sol' in file reloj-de-sol-digital.scad, line 175
Compiling design (CSG Primitive generation)

```

Figura 27.10: Assert impide que el usuario trate de horadar el reloj con rayos paralelos al horizonte.

A Cecilia, mientras contemplaba la figura 27.10, le gustó el hecho de que `assert` revelara también toda la cadena de llamadas entre funciones y módulos que condujeron al espantable error: en más de una oportunidad podría tratarse de información muy relevante.

Antonia miraba el monitor con un gesto indeciso:

—La condición está bien —admitió—; el rayo de Sol sólo puede crearse si `alfa!=0` y `alfa!=180`. Sin embargo, imagino que el lector de este texto no necesariamente esté interesado en el ángulo `alfa`: quizá piense más en términos de ‘horas’ y ‘minutos’, que son los que finalmente querrá ver expresados en el reloj. Lo que quiero decir es que cualquier información sobre el ángulo seguramente le resultará tan inexpressiva como la alusión original a los infinitos que, por defecto, lanzó OpenSCAD.

Cecilia comprendió el punto: el ángulo alfa no era más que un detalle de la implementación interna del algoritmo. Tras recorrer nuevamente el texto, decidió que el responsable de comprobar la validez de la hora era el módulo `hora_solar`, por lo que luego de deshacer la modificación introducida en `rayo_de_sol` compartió el siguiente texto con Antonia:

```
1 module hora_solar(horas,minutos){  
2     hora=horas+minutos/60;  
3     assert(hora!=6 && hora!=18,"La hora no debe ser  
4         las 6:00 ni las 18:00.");  
5     alfa=alfa(hora);  
6     hora_decenas=n_a_digito(horas,1);  
7     hora_unidades=n_a_digito(horas,0);  
8     minuto_decenas=n_a_digito(minutos,1);  
9     minuto_unidades=n_a_digito(minutos,0);  
10    // Varias lineas de texto  
11 }  
12 module reloj_de_sol(){  
13     difference(){  
14         cuerpo(largo_reloj);  
15         hora_solar(18,0);  
16     }  
17 }  
18 reloj_de_sol();
```

```
Compiling design (CSG Tree generation)...  
ERROR: Assertion `((hora != 6) && (hora != 18))' failed: "La hora no debe ser las 6:00 ni las 18:00." in file reloj-de-sol-digital.scad, line 126  
TRACE: called by 'assert' in file reloj-de-sol-digital.scad, line 126  
TRACE: called by 'hora_solar' in file reloj-de-sol-digital.scad, line 167  
TRACE: called by 'difference' in file reloj-de-sol-digital.scad, line 162  
TRACE: called by 'reloj_de_sol' in file reloj-de-sol-digital.scad, line 176  
Compling design (CSG Products generation)
```

Figura 27.11: Cecilia desplaza el `assert` al módulo `hora_solar`.

Como el valor de la hora debía usarse tantas veces, Cecilia decidió además realizar su cálculo al principio, en la línea 2. No sin recelo dirigió la mirada en dirección a su amiga: encontró en ella nuevamente una expresión no del todo satisfecha.

—¿Sabés qué? —adelantó Antonia—. Ya que estamos, podríamos impedir también la mera posibilidad de que algún lector descuidado pretenda agujerear el reloj ‘desde abajo’, pasando horas previas a las 6:00 o posteriores a las 18:00... ¿No te parece?

Cecilia suspiró con ostensible cansancio; mas debía reconocer que, muy a su pesar, la propuesta de Antonia era bastante razonable. En cualquier caso, implementar esa precaución no le pareció muy difícil, y confió en que ya le quedaran pocas páginas al capítulo.

```

1 module hora_solar(horas,minutos){
2     hora=horas+minutos/60;
3     assert(hora>6 && hora<18,"La hora debe
4         encontrarse entre las 6:00 y las 18:00.");
5     alfa=alfa(hora);
6     hora_decenas=n_a_dígito(horas,1);
7     hora_unidades=n_a_dígito(horas,0);
8     minuto_decenas=n_a_dígito(minutos,1);
9     minuto_unidades=n_a_dígito(minutos,0);
10    // Varias líneas de texto
11    module reloj_de_sol(){
12        difference(){
13            cuerpo(largo_reloj);
14            hora_solar(21,0);
15        }
16    }
17    reloj_de_sol();

```

```

Compiling design (CSG Tree generation)...
ERROR: Assertion '(hora > 6) && (hora < 18))' failed: "La hora debe encontrarse entre las 6:00 y las 18:00." in file reloj-de-sol-digital.scad, line 126
TRACE: called by 'assert' in file reloj-de-sol-digital.scad, line 126
TRACE: called by 'hora_solar' in file reloj-de-sol-digital.scad, line 165
TRACE: called by 'difference' in file reloj-de-sol-digital.scad, line 162
TRACE: called by 'reloj_de_sol' in file reloj-de-sol-digital.scad, line 173
Compiling design (CSG Products generation)...

```

Figura 27.12: De puro precavidas, Antonia y Cecilia deciden evitar incluso la posibilidad de que un usuario pretenda agujerear horas nocturnas.

Antonia, ahora sí, demostró su conformidad asintiendo con

una sonrisa.

—Ya sabemos cómo impedir las horas conflictivas —aprobó—; ahora deberíamos asegurarnos de que el resto es aceptado por el módulo `reloj_de_sol`.

Cecilia estuvo a punto de decir que sí, pero algo la inquietaba desde hacía rato, sin que pudiera definir muy bien qué. Tenía que ver con el error que acababan, presumiblemente al menos, de resolver; tenía que ver con la tangente...

—¡Pará! —lanzó Cecilia, tomando conciencia de pronto del problema que bullía en su inconsciente—. La función `tangente` no sólo puede devolver un conflictivo 0, sino un aún más problemático ‘infinito’... ¡Y de hecho lo hace para 90° ! ¡Nada menos que a pleno mediodía! —Cecilia sintió un escalofrío que la obnubiló e impidió recordar que varias veces habían probado ya esa hora meridiana; se lanzó así sobre el teclado casi con la seguridad de ver aparecer otro error:

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(12,0);
5     }
6 }
7 reloj_de_sol();
```

La tranquila aparición de las 12:00 en la figura 27.13 sólo contribuyó a confundirla más. Antonia tomó suavemente el teclado, mientras explicaba:

—Efectivamente, OpenSCAD sabe perfectamente que `tan(90)` no tiene solución en los números reales, y lo expresa a su modo:

»La cuestión es que si dividís un número real por ‘infinito’ en OpenSCAD, obtenés 0 —aclaró Antonia⁶—. Es... raro, lo admito; pero no podemos negar que resulta conveniente y hasta intuitivo,

⁶Que conste que me quedo sólo porque me lo pidieron! (Nota del Editor)

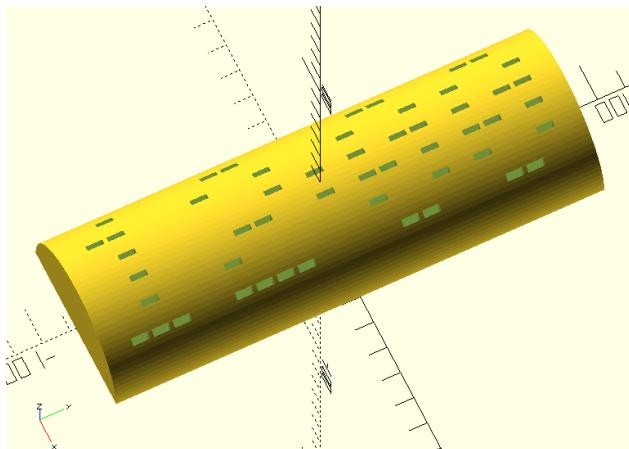


Figura 27.13: Las 12:00 son representadas apaciblemente por el módulo `reloj_de_sol` aun cuando la tangente de 90° no tenga solución en los números reales.

```
echo (tan(90));
```

Consola
 Compiling design (CSG Tree generation)...
 ECHO: inf
 Compiling design (CSG Products generation)

Figura 27.14: OpenSCAD adopta la convención de que $\tan 90^\circ = \infty$.

sobre todo en un dominio claramente más geométrico que aritmético —añadió, alzándose de hombros y buscando la complicidad de Cecilia, quien con un suspiro de alivio y alzando los ojos al techo expresó con claridad que estaba de acuerdo, quizá más por cansancio que por convicción algebraica.

```
echo (1/tan(90));
```

Consola
 Compiling design (CSG Tree generation)...
 ECHO: 0
 Compiling design (CSG Products generation)

Figura 27.15: Para escándalo de los matemáticos, en OpenSCAD $\frac{1}{\tan 90^\circ} = 0$.

—¿Te parece que dejemos la solución del otro problema para el capítulo que viene? —propuso Antonia, y a Cecilia nuevamente no le costó nada estar de acuerdo de todo corazón.

El reloj de Sol – II

LA MAÑANA siguiente Cecilia se levantó más temprano que de costumbre; recorría impaciente los interminables pasillos de Hardvard, pasando una y otra vez por delante de la puerta cerrada de la oficina de Antonia. Los primeros rayos de Sol ya atravesaban limpiamente los altos ventanales, trazando en el aire delicadas, euclídeas y quietas líneas luminosas. Cecilia las contemplaba y confirmaba su sospecha acerca de la causa del segundo problema que la desconcertó el día anterior.

Antonia dobló la esquina del largo pasillo que daba a su oficina; Cecilia, al verla, reprimió un grito de impaciencia y, apoyando los puños en su cintura, tamborileó ostensiblemente con un pie contra el piso mientras fruncía el gesto con aire burlón. Antonia, por supuesto, pareció no darse cuenta de la pequeña farsa urgente, porque mantuvo su paso tranquilo hasta la puerta de su oficina.

Una vez dentro, y mientras se sentaban frente a la computadora esperando que Linux preparara la diaria sesión, Cecilia compartió con Antonia su diagnóstico:

—¿Te acordás de que cuando quisimos agujerear el reloj con todas las horas desde las 6:00 hasta las 18:00 quitamos de él limpias rebanadas completas? —preguntó retóricamente, ya que estaba segura de que Antonia lo recordaba perfectamente—. Pues

bien; creo que ya se porqué ocurrió —y detuvo su discurso con una sonrisa satisfecha, procurando crear un clima de suspenso.

Antonia la contemplaba en silencio y devolviéndole una sonrisa serena, por lo que Cecilia, algo mortificada, decidió continuar:

—El giro aparente del Sol alrededor nuestro es muy lento; por esta razón, las horas que horadamos quedaron muy ‘pegaditas’ una a la otra, y terminaron por confundir sus cortes entre sí —y aprovechando que la computadora mostraba ya operativo el gestor de ventanas, escribió con rapidez los dos ejemplos de las figuras 28.1 y 28.2.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(13,0);
5     }
6 }
7 reloj_de_sol();
```

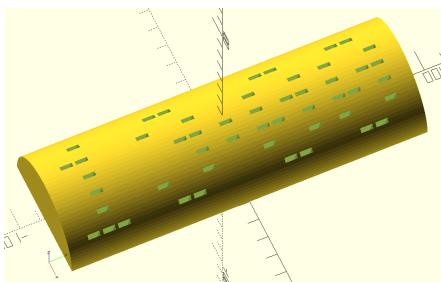


Figura 28.1: Cecilia pretende mostrar que las 13:00...

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(13,1);
5     }
6 }
7 reloj_de_sol();
```

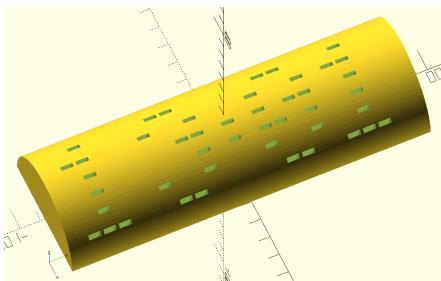


Figura 28.2: ...y las 13:01 se encuentran muy cercanas entre sí, y por eso se confunden sus cortes.

Cecilia no estaba segura de que su punto hubiera quedado

claro; de pronto sintió de manera particularmente intensa la inseguridad que Antonia debía padecer cuando tenía que explicar casi cualquier cosa.

—Si trato de marcar ambas horas a la vez, se van a confundir entre sí —aseguró ahora simulando aplomo mientras escribía otro ejemplo.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(13,0);
5         hora_solar(13,1);
6     }
7 }
8 reloj_de_sol();
```

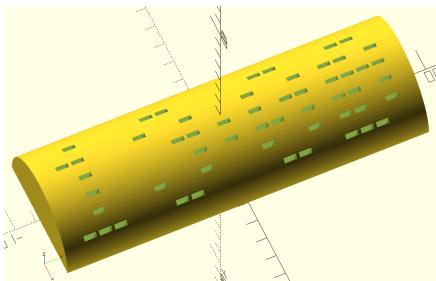


Figura 28.3: Cecilia muestra las 13:00 y las 13:01 a la vez, procurando que se aprecie con claridad la confusión que se produce entre ambas horas.

Antonia asintió ahora plenamente y con una amplia sonrisa. Cecilia no pudo descifrar si aprobaba su conclusión o la manera de exponerla.

—Creo que debemos resignarnos, entonces, a no indicar *todas* las horas, sino a ciertos intervalos que no se solapen entre sí —propuso Cecilia.

Antonia expresó su acuerdo marcando aún más su sonrisa; tomó el teclado para sí y lanzó un «¡Dale! ¡Probemos!» que resonó con entusiasmo en la oficina.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(13,0);
5         hora_solar(13,10);
```

```
6     }  
7 }  
8 reloj_de_sol();
```

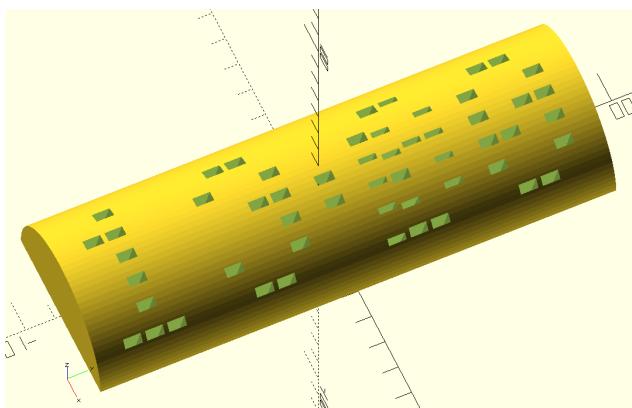


Figura 28.4: Antonia comprueba que las 13:00 y 13:10 se encuentran aún demasiado juntas.

—Hmmm... parece que 10 minutos sigue siendo un intervalo demasiado exiguo; probemos con... 20 —Antonia no disimulaba muy bien su sobreactuada incertidumbre; Cecilia recordó, además, que su amiga ya había resuelto e incluso impreso su propio reloj de Sol digital antes de comenzar el manual. Supuso que, impulsada por su invencible inclinación didáctica, trataba de recrear para ella los pasos que la condujeron a la solución. Cecilia no pudo menos que sentir una gran ternura.

```
1 module reloj_de_sol(){  
2     difference(){  
3         cuerpo(largo_reloj);  
4         hora_solar(13,0);  
5         hora_solar(13,20);  
6     }  
7 }
```

```
8 reloj_de_sol();
```

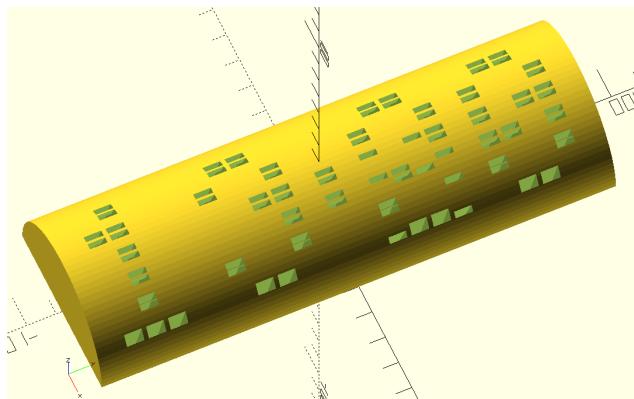


Figura 28.5: Las 13:00 y las 13:20 parecen distinguirse entre sí.

—Parece que 20 minutos es un intervalo conveniente —sugirió Antonia, y Cecilia no pudo estar en desacuerdo.

28.1. ROTACIÓN DEL CAMPO DE VISIÓN

—¿No te gustaría ver cómo queda cada hora desde la perspectiva del Sol? —inquirió Antonia—. Me refiero a poder mirar el reloj desde la dirección precisa de cada corte; de esa manera, podríamos confirmar que cada uno resulta bien distingible del otro.

Cecilia no estaba segura de comprender la pregunta de Antonia, pero en su tono percibió que era más bien retórica, por lo que con la mirada la invitó a continuar.

—Voy a tratar de rotar la vista... a ver... —Antonia se mordía levemente el labio inferior mientras trataba de hacer puntería con el ratón; evidentemente, lo suyo era la escritura, y no los dispositivos apuntadores.

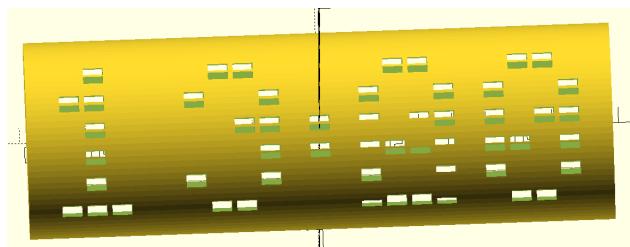


Figura 28.6: Antonia trata en vano de alinear la vista con el ratón.

»Por suerte hay una manera de dirigir la mirada por escrito —anunció Antonia con tono triunfal; Cecilia comprendió entonces que toda esa pantomima con el ratón no había sido otra cosa que uno más de sus trucos infantiles para introducir un tema nuevo, y exaltar de paso las virtudes de las palabras sobre los demás recursos expresivos.

»La variable del sistema \$vpr sirve para controlar desde dónde querés ver exactamente la escena; se trata de un vector que indica dicha dirección. Te confieso que a mí me resulta bastante difícil deducir, *a priori*, qué valor debe tener para mostrarme los objetos como quiero —admitío—; pero hay un truquito muy útil —agregó con un guiño—. Fijate abajo a la izquierda.

Cecilia dirigió su mirada a ese extremo del monitor.

```
Viewport: translate = [ -1.87 5.36 7.38 ] rotate = [ 15.70 0.00 87.90 ] dis
```

Figura 28.7: La variable \$vpr tiene el valor indicado por la etiqueta `rotate`.

—¿Viste dónde dice `rotate=[15.70 0.00 87.90]` en la figura 28.7? Ése es el valor que tiene, ahora, la variable \$vpr. Está claro que esos decimales son los responsables de la oblicuidad desproporcionada de la vista que logré, y a duras penas, con el ratón. Así que no resulta muy difícil deducir cuáles deberían ser los valores

'correctos'.

```
1 $vpr=[15,0,90];
2 reloj_de_sol();
```

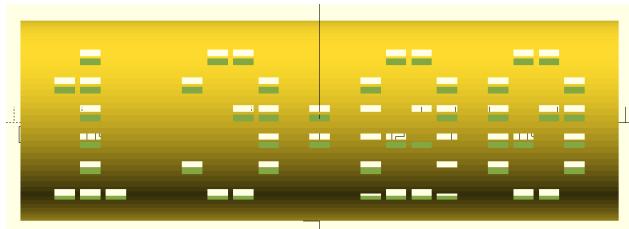


Figura 28.8: Reloj visto desde la dirección de los rayos solares a las 13:00.

A Cecilia le gustó mucho esa posibilidad:

—Ahora debemos reemplazar el '15' por otro ángulo para poder ver las 13:20 de frente, ¿no? —preguntó.

—Sí —confirmó Antonia, pero en su voz sonaba un dejo de insatisfacción; miraba el monitor como si algo la molestara—. Quizá te parezca un tanto quisquillosa, pero me parece que los ejes de la vista gráfica y los números sobre ellos escritos nos van a molestar un poco, al interferir con los pixeles. Es muy sutil, pero...

A Cecilia esas marcas no la molestaban en absoluto, pero desde hacía tiempo sabía que su amiga era muy quisquillosa, y la aceptaba como era.

Sin esperar su aprobación, Antonia continuó:

—Si desmarcás los botones correspondientes que están sobre la consola de mensajes, ejes y marcas desaparecerán.

Ahora, visiblemente más contenta frente a la figura 28.9, Antonia prosiguió:

—Como decías, ahora debemos cambiar el '15' por otro ángulo, pero en lugar de calcularlo vamos a dejar que OpenSCAD lo

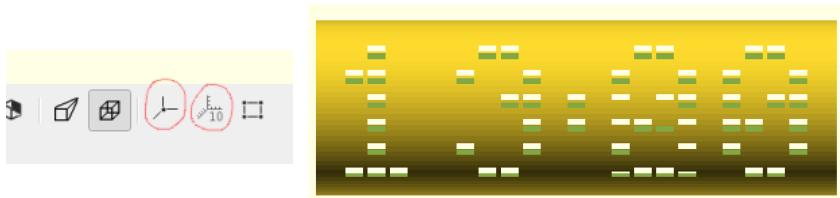


Figura 28.9: Los ejes de coordenadas y sus marcas pueden ser invisibilizados.

haga por nosotras —y acompañó esta propuesta con otro de sus infaltables guiños.

```
1 $vpr=[90-alfa(13+20/60),0,90];
2 reloj_de_sol();
```

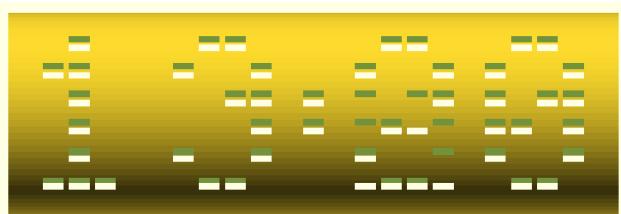


Figura 28.10: Reloj visto desde la dirección de los rayos solares a las 13:20.

Cecilia no pudo evitar ceder a la pulsión por comprender; con los ojos entornados observó atentamente el texto. Le pareció que el ‘90’ ubicado en la tercera posición del vector de \$vpr representaba la rotación de la vista alrededor del eje Z: quizás gracias a ella estaban viendo el reloj “horizontalmente” en relación al monitor. Por su parte, el valor que ocupaba el primer lugar en el vector tal vez determinaba la rotación en torno al eje X, que coincidía con el eje principal del reloj. Dicho ángulo se obtenía restando de 90 el que devolvía la función alfa que habían escrito, capítulos

antes, con Antonia: tal vez esa resta se debía a que ellas medían los ángulos desde las 12:00, mientras que OpenSCAD seguramente lo haría desde el plano XY.

Antonia sacó a Cecilia bruscamente de sus cavilaciones:

—¿No tenés ganas de agujerear más horas? —preguntó, mientras escribía.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(12,0);
5         hora_solar(12,20);
6         hora_solar(12,40);
7         hora_solar(13,0);
8         hora_solar(13,20);
9         hora_solar(13,40);
10    }
11 }
12 reloj_de_sol();
```

28.2. ANIMACIONES

Cecilia se sentía cada vez más entusiasmada y feliz. La compilación de las seis horas elegidas llevó su tiempo en la computadora de Antonia, pero le parecía razonable: se trataba de varios orificios. Por lo demás, después de 28 capítulos de pésima literatura y poco rigurosa exposición técnica se sentía perfectamente capaz de esperar unos cuantos minutos más.

—Ahora podríamos ver el reloj desde cada una de las horas perforadas —el tono de Antonia era inequívoco: indudablemente estaba por proponer algo que creía mucho mejor que eso—. Pero... ¿no te gustaría que OpenSCAD rotara animadamente el reloj ante tus ojos para que pudieras apreciar la suave transición de las horas?

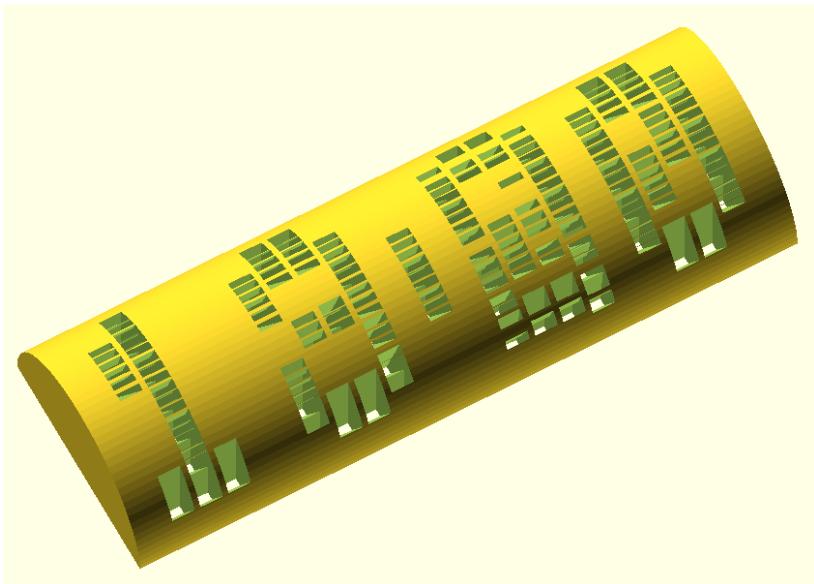


Figura 28.11: Reloj horadado con las horas que van desde las 12:00 a las 13:40 en intervalos de 20 minutos.

Cecilia los abrió desmesuradamente: ¿Sería posible? Inmediatamente comprendió que sí; a esta altura, ya nada la sorprendía del ingenio de los creadores de este lenguaje.

—Empecemos por un ejemplo... elemental —dijo Antonia, aunque Cecilia tradujo para sus adentros ese término con un seguramente más apropiado “tonto”.

```
1  rotate([0,0,$t*360])
2    cube([30,30,2],center=true);
3
4  translate([40,0,0])
5    rotate([0,0,-$t*360])
6      cube([20,20,2],center=true);
```

»Como podrás apreciar, en principio escribí dos simples cubos.

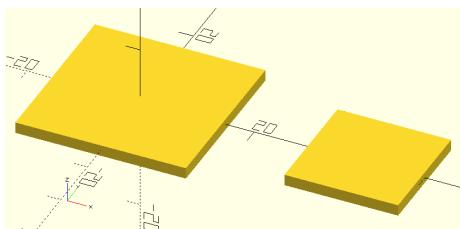


Figura 28.12: Cubos dispuestos para ser animados por OpenSCAD.

Bueno, está bien: ortoedros —aclaró a regañadientes—. Pero fijate que en las rotaciones de las líneas 1 y 5 aludo a una extraña variable aparentemente no definida: $\$t$ —Antonia puso cara de misterio antes de proseguir—. Y OpenSCAD ni chistó: la figura 28.12 parece indicar que la conoce.

Cecilia se preguntaba en silencio hasta cuándo su amiga iba a dilatar ineфicazmente el suspenso.

—Pues bien, resulta que $\$t$ es otra variable más del sistema. Representa el tiempo: ese gran misterio que deslumbró y sigue atareando a los más diversos filósofos y artistas; esa materia de la que, al decir de Borges, estamos hechos —el tono de Antonia adquirió, quizás a su pesar, un cierto tono vibrante. Cecilia hizo un esfuerzo por reprimir un bostezo.

—Si ahora hacés *click* en la entrada del menú Ver → Animar, vas a notar que sobre la consola aparecen mágicamente tres nuevos cuadros de texto: Tiempo, FPS y Pasos.

»En el cuadro Tiempo no escribas nada; su función exclusiva consiste en mostrar el avance del mismo, de 0 a 1 y vuelta a empezar desde 0, cíclicamente, en intervalos de 0,01 —explicó Antonia y se detuvo un momento, mientras dirigía una mirada ensueñadora al techo—. Las hermosas paradojas de Zenón de Elea no hubieran alarmado nuestra juventud si el tiempo fuera tan ostensiblemente granular —y suspiró con una melancólica sonrisa.

»Los cuadros para los que sí es necesario decidir valores son los

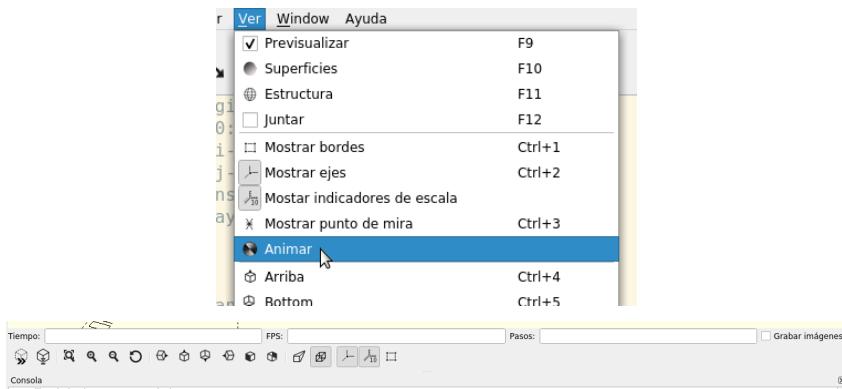


Figura 28.13: El menú **Animar** permite la animación de las escenas de OpenSCAD.

otros dos —Antonia retomó no sin esfuerzo su explicación—. En FPS debés indicar la velocidad con la que querés que se desarrolle la animación (FPS, como sin duda sabrás, son las siglas de *Frames Per Second*: cuadros por segundo, en inglés): un valor más alto representa una velocidad mayor. Por su parte, en Pasos debés señalar en cuantos ‘cuadros’ (*Frames*) querés dividir el intervalo de tiempo completo de 0 a 1. Por ejemplo, si ponemos 5 en FPS y 100 en Pasos, la animación discurrirá a una velocidad de 5 cuadros por segundo (una suerte de ‘cámara lenta’) y la rotación completa se realizará por saltos de $3,6^\circ$ ($360^\circ / 100$), porque la rotación avanza según $t * 360$: fijate las líneas 1 y 5 del ejemplo.

A Cecilia le pareció mucha información junta, y la explicación de Antonia —como siempre— no era de la mayor eficacia; decidió una vez más que la mejor manera de aprehender este nuevo concepto sería probarlo y escribirlo varias veces por su propia cuenta en ejemplos diversos.

—Si en lugar de cubos hubiéramos escrito engranajes, la animación sería bastante más interesante —reflexionó Antonia, con

un dejo de resignación—. Pero en fin; veamos cómo aplicar las animaciones a la rotación del campo de visión.

»En principio y en nuestro ejemplo, queremos que la rotación abarque desde las 12:00 hasta las 13:40. Por un lado, para mirar de frente a las 12:00, \$vpr debe valer [0,0,90], y para las 13:40 debe ser igual a [25,0,90], porque $90 - \alpha(13:40/60) = 25$ —dijo, mientras escribía los ejemplos de las figuras 28.14 y 28.15.

```
1 $vpr=[0,0,90];
2 reloj_de_sol();
```

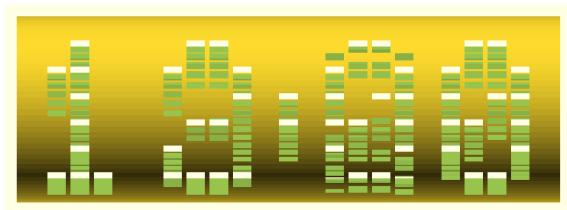


Figura 28.14: Para ver de frente las 12:00, \$vpr=[0,0,90].

```
1 $vpr=[25,0,90];
2 reloj_de_sol();
```

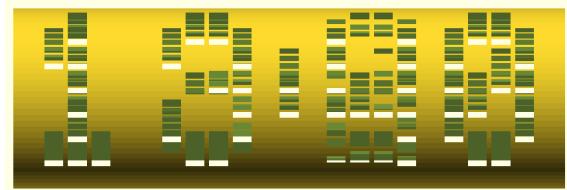


Figura 28.15: Para ver de frente las 13:40, \$vpr=[25,0,90].

»Así que necesitamos que para $t=0$ sea $$vpr=[0,0,90]$, y para $t=1$ sea $$vpr=[25,0,90]$. Creo que la conclusión es inmediata —expresó Antonia con su invencible optimismo.

```
1 $vpr=[$t*25,0,90];
2 reloj_de_sol();
```

Antonia empleó 2 y 40 para FPS y Pasos, respectivamente. A Cecilia le gustó la animación, pero algo la inquietó súbitamente:

—Antonia... en la imagen 28.15 puedo ver algunos pixeles ‘medio encendidos’ debajo del 4...

Antonia frunció los labios mientras asentía levemente:

—Sí... es verdad —admitió—. Es un problema. Para evitarlo se me ocurren dos caminos: hacer los pixeles menos altos, o separar más las horas entre sí. Ninguna de las soluciones me satisface del todo: señalar las horas cada 30 minutos relega nuestro reloj a un estado casi testimonial, por así decirlo; y pixeles muy delgados podrían atentar contra la legibilidad de la hora.

Tras unos instantes en que ambas amigas dudaban frente al monitor, evaluando incluso la posibilidad de dejar todo como estaba, Cecilia se decidió a probar la primera opción:

```
alto_pixel = 1.6; // era 2
```

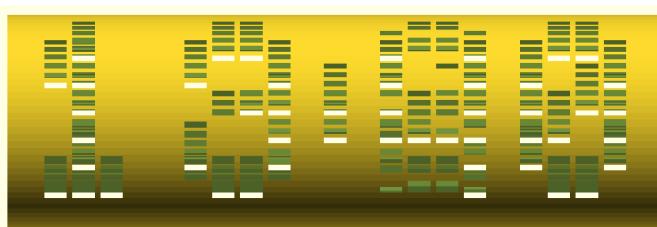


Figura 28.16: Cecilia prueba la posibilidad de que $\text{alto_pixel}=1.6$.

—No está mal; yo diría que lo dejemos así —propuso Antonia contemplando la figura 28.16—. Y con respecto a la animación, a los lectores se las debo; incrustar un gif animado en un pdf no siempre funciona, y en todo caso torna muy grande el archivo resultante —se excusó.

—¿Y no te parece una pena que no la puedan ver..? —preguntó Cecilia.

Antonia se encogió de hombros:

—Sinceramente, no puedo imaginar un lector de este manual que haya llegado a la página 282 y no tenga OpenSCAD a la mano — afirmó, y casi inmediatamente, añadió—: De hecho, casi no puedo imaginar ningún tipo de lector para este manual.

Cecilia no pudo evitar estar de acuerdo.

El reloj de Sol – III

La optimización prematura es la raíz de todos los males en computación.¹

EL SOL, ASOMADO a la ventana de la oficina de Antonia, encontró bien temprano a nuestras protagonistas sentadas frente a la siempre dócil computadora.

Cecilia se sentía tan cerca de terminar el reloj que casi no era capaz de disfrutar de la felicidad que —no lo dudaba— debía embargarla. Su estado le recordaba las ocasiones en que se encontraba “pasada de sueño” después de una larga noche en vela, al extremo de no sentir ya ganas de dormir.

—Ahora sí... ¡Atravesemos nuestro reloj con todas las horas del día! —exclamó, procurando sonar festiva y radiante.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(6,20);
5         hora_solar(6,40);
6         for(hora=[7:17],
```

¹Atribuido a Donald Knuth, C.A.R. Hoare y Edsger Dijkstra hacia 1970 (aunque probablemente haya sido parte del folclor informático desde siempre).

```
7         minuto=[0,20,40])
8             hora_solar(hora,minuto);
9     }
10 }
11 reloj_de_sol();
```

Cecilia releyó su texto no sin orgullo. Debía horadar el reloj con las horas comprendidas entre las 6:20 y las 17:40, a intervalos de 20 minutos. Para eso creó un bucle doble en las líneas 6 y 7: el primero recorría las horas desde las 7 hasta las 17 mientras que el segundo debía ofrecer los minutos 0, 20 y 40, por lo que no tenía sentido escribir un rango: con un simple vector explícito bastaba. Las 6:20 y las 6:40 hubieran sido difíciles de incluir en el bucle: consideró más económico expresarlas por separado en las líneas 4 y 5.

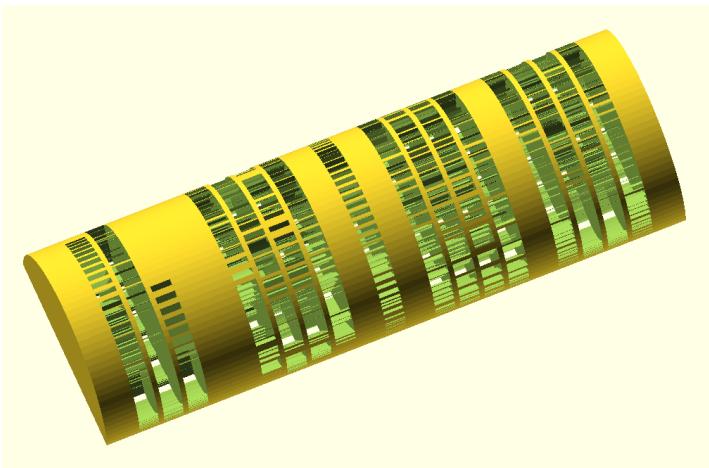


Figura 29.1: Primera versión del reloj de Sol digital.

La vista previa, con , fue prácticamente instantánea: en la computadora de Antonia demandó apenas 0,166 segundos. Pero Cecilia sabía que para poder imprimir el reloj era necesario

compilarlo con la más exigente : sólo tras casi 10 minutos, y con los ventiladores del procesador aullando a toda velocidad, el Intel i7 de 8.^{va} generación con 32 GB de RAM fue capaz de desplegar finalmente frente a sus ojos el tan largamente ansiado reloj.

Cecilia, una vez más a lo largo de este manual, giró sobre sí misma para compartir con Antonia su emoción, que ahora sí se abría paso sensiblemente en su pecho a la vista del reloj terminado. Pero, como también ocurriera tantas otras veces, encontró a Antonia con un gesto de reserva y contemplando con ojos críticos el resultado y el texto. Cecilia supo inmediatamente que este capítulo distaba mucho de estar concluido:

—¿Y ahora qué? —preguntó lacónicamente y con desgano.

Antonia salió rápidamente de su ensimismamiento:

—¡No te pongas así! —bromeó, riendo—. Por supuesto que así como está podemos darnos por satisfechas, y mandarlo a imprimir. Sin embargo... ¿Puede una artista dar alguna vez por concluida su obra? —preguntó con aire retórico.

Cecilia no se sentía una artista: sólo quería terminar el reloj. Además, recordó con terror que algunos pintores llegaron a afirmar que vendían sus obras sólo para no pasarse la vida retocándolas: ¿Pretendería Antonia que este manual durara tanto tiempo? Porque no se imaginaba vendiendo relojes de Sol digitales... Tras un instante, decidió sin embargo desechar esos temores y confiar en su amiga como hasta ahora, por lo que con la mirada la invitó a continuar.

29.1. OPTIMIZACIÓN

Antonia parecía buscar la mejor manera de comenzar; tal vez cediendo a su pulsión docente, lo hizo una vez más con una pregunta:

—¿Cuántos dígitos aparecen en la posición de las unidades de los minutos?

Cecilia se sintió obligada a responder:

—Pues... uno solo: el cero, por supuesto —y ni bien terminó la frase, sintió que un asomo de comprensión empezaba a iluminarla de manera difusa pero clara.

—O sea que desde las 6:20 hasta las 17:40 ese último dígito será siempre y constantemente '0', ¿no? —era evidente que Antonia trataba de dotar estas preguntas de un tono de neutra ingenuidad.

—Claro... —respondió lentamente Cecilia, procurando con todas sus fuerzas hacer surgir y dar cuerpo a la incipiente idea que pugnaba por aflorar en su conciente.

—¡Qué lástima tener que usar tantos rayos para un sólo dígito repetido...! —el lamento de Antonia no podía ser más sobreactuado; Cecilia casi empezaba a ponerse nerviosa:

—¿Proponés que atravesemos el cero con un solo rayo? —preguntó, más para ver si conseguía de Antonia que revelara su secreto que por otra razón.

Antonia, por toda respuesta, simplemente se alzó de hombros y simuló un gesto de perplejidad poco convincente.

Cecilia, tras unos momentos de intensa reflexión con la mirada vuelta hacia el monitor, finalmente se dio por vencida:

—Ok, Antonia... ¿Qué es lo que proponés?

29.2. UN HAZ DE RAYOS DE SOL

Antonia tomó el teclado con resignación; era evidente que le hubiera gustado que Cecilia resolviera por su cuenta el enigma. Como tantos otros docentes, era injusta: pretendía que los demás resolvieran en pocos segundos cuestiones que a ellos les habían demandado semanas, cuando no meses.

—¿Te acordás cuando empezamos a escribir nuestro rayo de

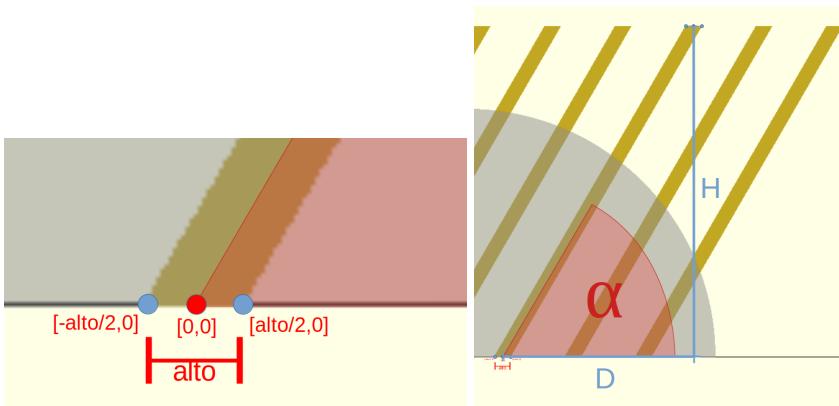


Figura 29.2: Gráficos auxiliares para la determinación de los extremos de cada rayo de Sol.

Sol, allá por el capítulo 18? —evocó, mientras desplegaba en el monitor los diagramas que copiamos en la figura 29.2.

Cecilia lo recordaba perfectamente: esos diagramas les habían permitido expresar las coordenadas de los vértices de cada rayo: $[-\text{alto}/2, 0]$, $[\text{D}-\text{alto}/2, \text{H}]$, $[\text{D}+\text{alto}/2, \text{H}]$, $[\text{alto}/2, 0]$.

—Supongamos que sabemos que un mismo dígito existe entre dos horas distintas, que se corresponden a dos ángulos diferentes: digamos, α_1 y α_2 —propuso cautamente Antonia, señalando la figura 29.3.

»¿Deberemos trazar *tooodos* los rayos entre ambos extremos..?— la manera en que arrastró las oes era una de las peores costumbres de Antonia como docente, pero a Cecilia casi no la molestó: ya sentía que la comprensión se abría paso con más seguridad en su mente.

—¿Vos decís que reemplazemos todos los rayos por un único haz? —aventuró Cecilia.

—¡Sí! —gritó triunfalmente Antonia, queriendo creer que su amiga ya había desentrañado el misterio del todo—. Fijate que

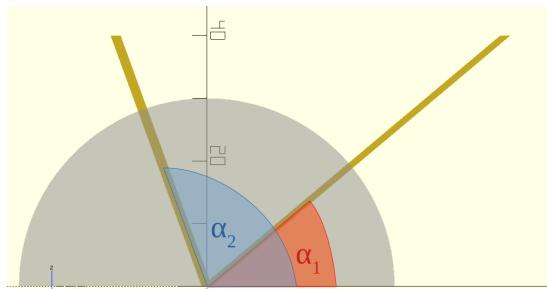


Figura 29.3: Dos rayos solares con distintas inclinaciones, que Antonia propone que pueden ser los extremos de un mismo dígito presente entre dos horas distintas.

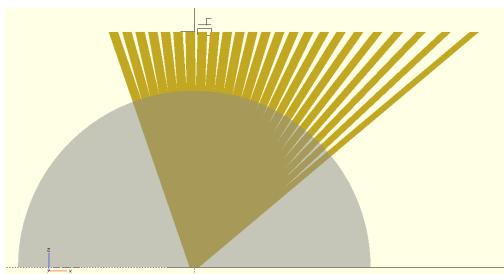


Figura 29.4: Antonia despliega los muchos rayos que deberían corresponderse con un sólo dígito que persistiera entre dos horas distintas.

cada rayo lo creamos como un polígono. Si bien nosotras venimos trabajando con ellos como si fueran necesariamente paralelogramos, nada cuesta reemplazar los vértices superiores de un solo rayo para convertirlo en un suficiente haz.

A Cecilia le gustó la propuesta expresada en la figura 29.5, y la idea —al menos como aspiración— le parecía cada vez más hermosa, si bien debía admitir para sus adentros que todavía no acababa de entenderla del todo.

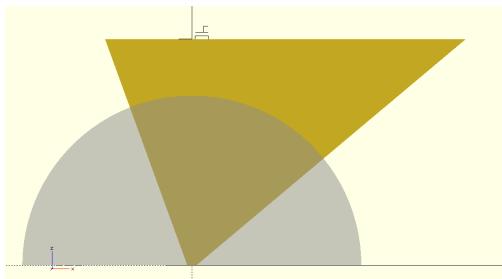


Figura 29.5: Haz de rayos de Sol, desplegados entre dos horas extremas.

—Para calcular los vértices superiores del haz que buscamos sólo debemos remitirnos a los correspondientes de los rayos extremos —siguió Antonia, quizá con demasiada velocidad. Ése era otro de sus vicios docentes preferidos: pasaba de una lentitud exasperante a una velocidad vertiginosa sin demostrar piedad alguna por sus circunstanciales alumnos.

Cecilia no pudo aguantar más:

—¡Pará! ¡Déjame pensar! —prorrumpió, y Antonia hizo silencio bruscamente. Revisó las páginas anteriores junto a la nueva figura 29.6; comprobó que los vértices de la base del nuevo haz seguían siendo $[-\text{alto}/2, 0]$ y $[\text{alto}/2, 0]$. Por su parte, los vértices superiores debían coincidir con los extremos de los dos rayos exteriores del haz: $[D_2 - \text{alto}/2, H]$ y $[D_1 + \text{alto}/2, H]$. D₁ y D₂ sabía cómo calcularlos: el primero era igual a $\frac{H}{\tan(\alpha_1)}$ y el segundo, analógicamente, a $\frac{H}{\tan(\alpha_2)}$.

Sin demasiada seguridad, debida seguramente al cansancio inducido por la tortuosa explicación de Antonia más que a cualquier otra cosa, buscó confirmación con una pregunta tentativa:

—¿Puede ser que el polígono del haz completo se defina con el vector $[[-\text{alto}/2, 0], [D_2 - \text{alto}/2, H], [D_1 + \text{alto}/2, H], [\text{alto}/2, 0]]$?

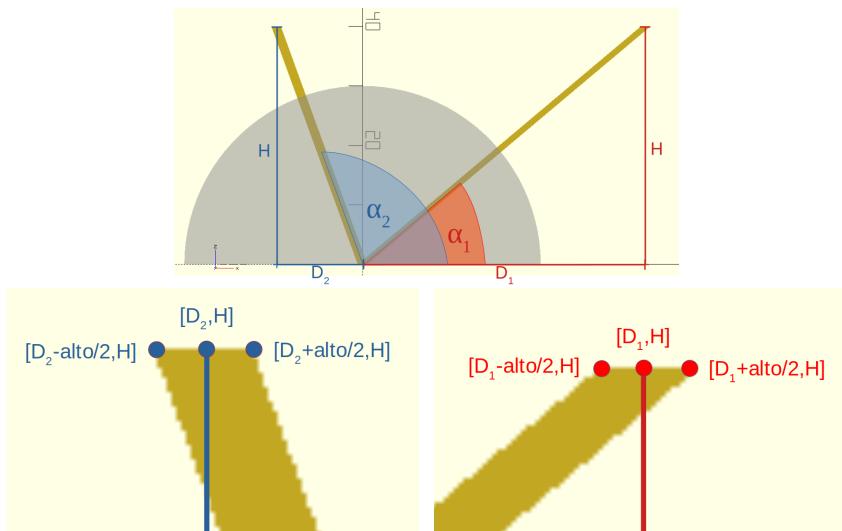


Figura 29.6: Coordenadas superiores del haz de rayos de Sol.

—¡¡¡Sí!!! —gritó Antonia, y hasta Fumington debió haberse enterado.

29.3. EL RAYO DE SOL REVISITADO

Cecilia volvió su mirada sobre el módulo `rayo_de_sol` original:

```

1 module rayo_de_sol(alfa){
2     D=H/tan(alfa);
3     vertices=[[ -alto_pixel/2,0],
4                [D-alto_pixel/2,H],
5                [D+alto_pixel/2,H],
6                [alto_pixel/2,0]];
7     // TODO: medir la duracion de esta solucion
8     //       y la de esta otra:
9     //       translate([0,ancho_pixel/2,0])
10    //       ojo : borrar 'center=true' abajo

```

```

11     rotate ([90,0,0])
12     linear_extrude(ancho_pixel,center=true)
13     polygon(vertices);
14 }
```

No le pareció muy difícil introducir los nuevos cambios, al tiempo que modificaba el nombre del módulo:

```

1 module haz_de_sol(alfa1,alfa2){
2   D1=H/tan(alfa1);
3   D2=H/tan(alfa2);
4   vertices=[[-alto_pixel/2,0] ,
5             [D2-alto_pixel/2,H] ,
6             [D1+alto_pixel/2,H] ,
7             [alto_pixel/2,0]];
8   // TODO: medir la duracion de esta solucion
9   //       y la de esta otra:
10  //       translate([0,ancho_pixel/2,0])
11  //       Ojo : borrar 'center=true' abajo
12  rotate ([90,0,0])
13  linear_extrude(ancho_pixel,center=true)
14  polygon(vertices);
15 }
```

En la línea 1 dejó lugar para recibir dos ángulos: α_1 y α_2 . En las líneas 2 y 3 calculaba, con ellos, D_1 y D_2 . Por último, los vértices de los extremos superiores quedaban expresados en las líneas 5 y 6. Lo demás, para su sorpresa, quedaba intacto. Aunque, claro, inmediatamente se dio cuenta de que el cambio introducido en este módulo impactaría en todos los que se basaban en él: es decir, prácticamente en todos.

—¿Ahora debemos modificar también el módulo `digito`, no?
—preguntó con la voz quebrada: ya se veía reescribiendo todo el texto completo.

—Sí—respondió Antonia procurando sonar tranquilizadora—; pero no te preocupes: verás que todos los cambios serán así de sencillos. La lógica que pusimos por escrito en estos módulos está

esencialmente bien; sólo debemos ampliarla un poco —y tras un instante, pareció querer aprovechar la ocasión para otra de sus apologías—. Esto es algo que comprobarás que te ocurre seguido —por no decir siempre— al escribir un texto un poco complejo: cuando lo termines advertirás que sólo arribaste a una primera versión, que necesita —y merece!— una reescritura. Y eso se debe fundamentalmente a que una nunca escribe un problema cuando lo entiende, sino que lo entiende cuando lo escribe.

Antonia se detuvo un momento, sin duda para dar tiempo a Cecilia a saborear la paradoja que sentía que acababa de ofrecer, pero también como si buscara la mejor manera de continuar, o tratara de evocar y resumir todos sus recuerdos al respecto:

—Lo que quiero decir es que una de las cosas más lindas de este género literario es que te ayuda a pensar; aclara tus ideas, o incluso las crea directamente. Una no escribe para poner en letras de molde sus ideas: una escribe para pensar, para *tener* ideas. Por eso la primera escritura de un código nunca puede ser definitiva: es siempre un primer borrador.

A Cecilia este panegírico no le pareció insensato: ella también había podido notar, a medida que escribía sus monografías de investigación astronómica, que las ideas sobre el tema que estudiaba se aclaraban e incluso surgían a expensas precisamente de la escritura. De pronto se sintió aún más cerca de Antonia que antes.

29.4. EL DÍGITO SOLAR REVISITADO

Cecilia volvió ahora sobre el módulo `digito` original:

```
1 module digito(alfa,numero){  
2     for(i=[0:5],j=[0:3]){  
3         digito=digitos[numero];  
4         if(digito[i][j]==1){  
5             x=(i-2.5)*(alto_pixel+delta_alto);
```

```

6      y=(j-1.5)*(ancho_pixel+delta_ancho);
7      translate([x,y,-0.01])
8      rayo_de_sol(alfa);
9  }
10 }
11 }

```

Después de leerlo varias veces, casi no pudo creer que sólo debiera introducir dos ligerísimos cambios:

```

1 module digito(numero,alfa1,alfa2){
2   for(i=[0:5],j=[0:3]){
3     digito=digitos[numero];
4     if(digito[i][j]==1){
5       x=(i-2.5)*(alto_pixel+delta_alto);
6       y=(j-1.5)*(ancho_pixel+delta_ancho);
7       translate([x,y,-0.01])
8       haz_de_sol(alfa1,alfa2);
9     }
10   }
11 }

```

Las modificaciones se limitaban a las líneas 1 y 8: la única responsabilidad del módulo consistía en elegir los píxeles a horadar y sólo necesitaba los ángulos para pasárselos al módulo `haz_de_sol`.

29.5. LA HORA SOLAR REVISITADA

El optimismo de Cecilia iba en aumento; con renovada confianza se lanzó sobre el módulo `hora_solar`:

```

1 module hora_solar(horas,minutos){
2   hora=horas+minutos/60;
3   assert(hora>6 && hora<18,"La hora debe
4     encontrarse entre las 6:00 y las 18:00.");
5   alfa=alfa(hora);
6   hora_decenas=n_a_dígito(horas,1);
7   hora_unidades=n_a_dígito(horas,0);

```

```
7   minuto_decenas=n_a_digito(minutos,1);
8   minuto_unidades=n_a_digito(minutos,0);
9   delta_y=ancho_pixel+delta_ancho;
10  // horas
11  if (hora_decenas != 0)
12    translate([0,-8.5*delta_y,0])
13      digito(alfa,hora_decenas);
14  translate([0,-3.5*delta_y,0])
15      digito(alfa,hora_unidades);
16  // minutos
17  translate([0,3.5*delta_y,0])
18      digito(alfa,minuto_decenas);
19  translate([0,8.5*delta_y,0])
20      digito(alfa,minuto_unidades);
21  // separador
22  separador(horas,minutos);
23 }
```

A medida que lo recorría con la mirada, su sonrisa fue congelándose hasta convertirse en una mueca de desconcierto. No tardó mucho en descubrir que había tropezado con un problema fundamental:

—Antonia, este módulo se basa en la idea de perforar el reloj con una hora *completa*: las 12:00, las 9:40, o la que sea. Pero nosotras ahora decidimos que sería mejor escribir el cero final de los minutos de una sola vez, desde las 6:20 a las 17:40, mientras el resto de los dígitos avanza por su cuenta —y al tiempo que decía esto, percibió que el problema se diversificaba aún más—: de hecho, *cada dígito* avanza a su propio y distinto ritmo: las decenas de los minutos lo hacen cada 20, las unidades de las horas cada hora, y las decenas de las horas sólo toman un valor: el ‘1’, y entre las 10:00 y las 17:40.

Tras unos instantes en los que buscó inútilmente en el gesto de su amiga alguna pista, preguntó alarmada:

—¿Debemos deshacernos de este módulo enteramente?

Antonia sonrió:

—Yo creo que no: me parece que es un lindo detalle permitir al eventual lector o lectora de este texto que imprima un reloj donde sólo consten algunas horas puntuales que le resulten significativas. Así, podría tener un reloj que sólo indique las 9:37, 11:41 y 16:22, por ejemplo. Imaginemos que, tal vez, esas horas particulares tengan un sentido especial para él o para ella.

A Cecilia le pareció una posibilidad muy extravagante; pero supuso también que un módulo más en el texto no podía arruinarlo ni abarrotar el espacio de ningún dispositivo de almacenamiento, por lo que podían en todo caso convivir con él.

—Las modificaciones a este módulo, entonces, son elementales también —anticipó Antonia, tomando ahora para sí el teclado.

```

1 module hora_solar(horas,minutos){
2     hora=horas+minutos/60;
3     assert(hora>6 && hora<18,"La hora debe
4         encontrarse entre las 6:00 y las 18:00.");
5     alfa=alfa(hora);
6     hora_decenas=n_a_dígito(horas,1);
7     hora_unidades=n_a_dígito(horas,0);
8     minuto_decenas=n_a_dígito(minutos,1);
9     minuto_unidades=n_a_dígito(minutos,0);
10    delta_y=ancho_pixel+delta_ancho;
11    // horas
12    if (hora_decenas != 0)
13        translate([0,-8.5*delta_y,0])
14            dígito(hora_decenas,alfa,alfa);
15    translate([0,-3.5*delta_y,0])
16        dígito(hora_unidades,alfa,alfa);
17    // minutos
18    translate([0,3.5*delta_y,0])
19        dígito(minuto_decenas,alfa,alfa);
20    translate([0,8.5*delta_y,0])
21        dígito(minuto_unidades,alfa,alfa);
22    // separador

```

```
22     separador(hora,minutos);  
23 }
```

Cecilia comprobó inmediatamente que todo se limitaba a crear —en las líneas 13, 15, 18 y 20— dígitos cuyos ángulos inicial y final coincidieran: de esa manera, el haz se convertía naturalmente en el rayo original. Sonrió con una extraña satisfacción: evidentemente, la lógica que habían conquistado en un principio seguía funcionando; un rayo de Sol era, apenas, un caso particular de haz en el que los ángulos extremos coincidían. Cecilia recuperó de pronto su buen humor.

—Eso sí: fijate en la línea 22. Debemos modificar también el módulo `separador` —advirtió Antonia, mientras lo señalaba con el mentón.

29.6. EL SEPARADOR REVISITADO

```
1 module separador(horas,minutos){  
2     alfa=alfa(horas+minutos/60);  
3     for(i=[-1,1])  
4         translate([i*0.5*(alto_pixel+delta_alto), 0,  
           -.01])  
5         rayo_de_sol(alfa);  
6 }
```

Cecilia estaba dispuesta a realizar otro cambio trivial, cuando advirtió que la cosa no era tan sencilla. El módulo esperaba como parámetros la hora y los minutos: eso funcionaba muy bien para un separador *instantáneo*, pero no para un separador que tuviera que extenderse entre dos horarios distintos. Para dar cuerpo a sus ideas, escribió una rápida modificación.

```
1 module separador(hora_inicial,  
2                   minutos_inicial,  
3                   hora_final,
```

```
4           minutos_final){
5     alfa1=alfa(hora_inicial+minutos_inicial/60);
6     alfa2=alfa(hora_final+minutos_final/60);
7     for(i=[-1,1])
8       translate([i*0.5*(alto_pixel+delta_alto), 0,
9                  -.01])
10      haz_de_sol(alfa1,alfa2);
11  }
```

—¿Qué te parece así? —preguntó, buscando la opinión de su amiga.

Antonia no parecía muy convencida:

—Funciona, por supuesto. Lo que no me convence del todo es que los dígitos reciben como parámetros α_1 y α_2 , calculados previamente por otro módulo a partir de las horas correspondientes: algo me dice que el separador debería adherir a la misma convención —y escribió su propia versión.

```
1 module separador(alfa1,alfa2){
2   for(i=[-1,1])
3     translate([i*0.5*(alto_pixel+delta_alto), 0,
4                -.01])
5   haz_de_sol(alfa1,alfa2);
6 }
```

»En principio, de esta manera el módulo hora_solar es fácil de adaptar; sólo hay que cambiar la línea 22 —agregó Antonia.

```
1 module hora_solar(horas,minutos){
2   hora=horas+minutos/60;
3   assert(hora>6 && hora<18,"La hora debe
4         encontrarse entre las 6:00 y las 18:00.");
5   alfa=alfa(hora);
6   hora_decenas=n_a_dígito(horas,1);
7   hora_unidades=n_a_dígito(horas,0);
8   minuto_decenas=n_a_dígito(minutos,1);
9   minuto_unidades=n_a_dígito(minutos,0);
10  delta_y=ancho_pixel+delta_ancho;
```

```
10 // horas
11 if (hora_decenas != 0)
12     translate([0,-8.5*delta_y,0])
13     digito(hora_decenas,alfa,alfa);
14     translate([0,-3.5*delta_y,0])
15     digito(hora_unidades,alfa,alfa);
16 // minutos
17     translate([0,3.5*delta_y,0])
18     digito(minuto_decenas,alfa,alfa);
19     translate([0,8.5*delta_y,0])
20     digito(minuto_unidades,alfa,alfa);
21 // separador
22 separador(alfa,alfa);
23 }
```

A Cecilia no le costó nada estar de acuerdo.

—Muy bien —resumió Antonia con una sonrisa satisfecha, y reclinándose contra la silla mientras contemplaba el texto sin disimular una nota de orgullo en su voz—. Acabamos de lograr algo muy importante: reescribimos el código original de forma más general, sin que por eso deje de funcionar en el caso particular que veníamos trabajando: horas y minutos puntuales.

```
1 module reloj_de_sol(){
2     difference(){
3         cuerpo(largo_reloj);
4         hora_solar(12,0);
5         hora_solar(15,40);
6     }
7 }
8 reloj_de_sol();
```

—Si no me equivoco, los angloparlantes llaman a este proceder *refactoring*² —agregó Antonia.

²Tiene equivalencia en castellano, y no suena tan mal: *refactorización*. (nota del Editor)

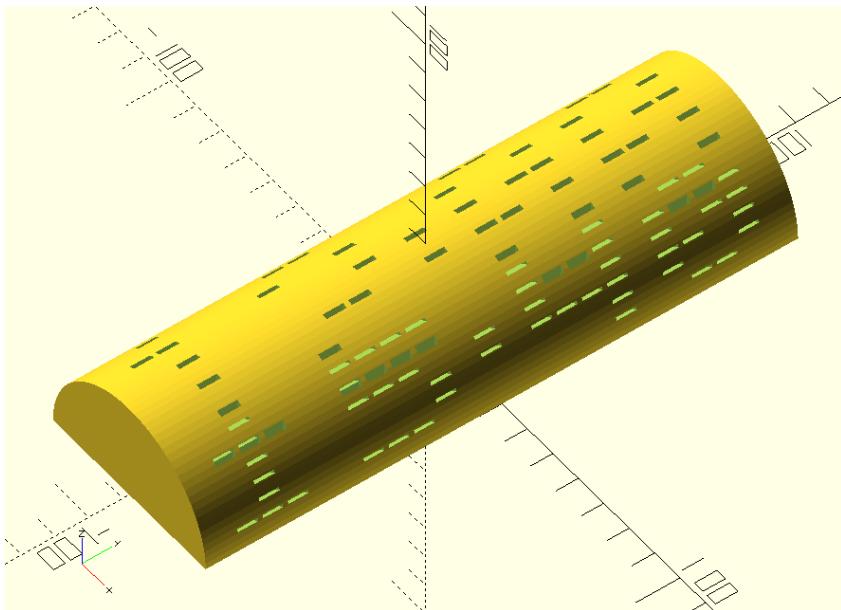


Figura 29.7: Dos horas puntuales logradas con el nuevo texto optimizado, lo que demuestra que la refactorización del mismo no afectó su funcionalidad original.

—¿Me prometés que en el capítulo siguiente terminamos el reloj? —suplicó Cecilia, que tras 17 páginas ya no quería otra cosa que descansar.

—Te lo prometo —concedió su amiga—. Al menos la cuestión central de las horas y los minutos. En el capítulo 31 quiero mostrarte cómo agregar texto tridimensional a los objetos: un reloj de Sol sin leyendas no es un reloj de Sol.

Cecilia recordaba esa otra promesa, y esperaba que esta vez Antonia cumpliera ambas a la brevedad.

— 30 —

El reloj de Sol – IV

A L DÍA SIGUIENTE Cecilia encontró a Antonia en su oficina ya ubicada frente a la computadora; se sentó a su lado, decidida a no levantarse sin haber terminado de agujearar todas las horas de manera óptima y final.

—En el capítulo anterior nos propusimos permitir al lector crear dos tipos de relojes de Sol: uno con algunas cuantas horas específicas, y otro presentando todas las que abarquen desde las 6:20 hasta las 17:40 —recordó Antonia—. Creo que cada uno se merece su propio módulo —y una vez más arrancó del teclado felices chasquidos.

30.1. EL RELOJ DE SOL DISCRETO

```
1 /* Reloj de Sol de algunas horas puntuales.
2   'vector_horas' es un vector con las horas
3   y minutos que el usuario desea mostrar;
4   por ej.: [[12,00], [7,13], [16,23]]
5   representa las 12:00, 7:13 y 16:23.
6 */
7 module reloj_de_sol_discreto(vector_horas){
8     difference(){
9         cuerpo(largo_reloj);
```

```
10     for(hora_minutos=vector_horas){
11         hora=hora_minutos[0];
12         minutos=hora_minutos[1];
13         hora_solar(hora,minutos);
14     }
15 }
16 }
17 reloj_de_sol_discreto([[12,00],[7,13],[16,23]]);
```

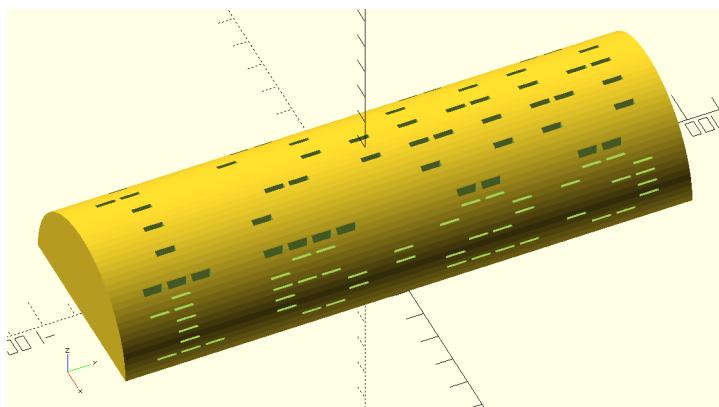


Figura 30.1: Reloj de Sol discreto, indicando unas cuantas horas puntuales.

—Cuando un comentario abarca varias líneas es más cómodo encerrarlo entre las marcas ‘/*’ y ‘*/’ —aclaró Antonia—. Después de todo, creo que este módulo merece un comentario de alguna extensión: es uno de los dos con los que se relacionará de manera más directa el lector de este texto. Los demás son en cierta manera ‘auxiliares’ —agregó, como justificándose.

A Cecilia le pareció muy apropiado, y dedicó su atención al módulo: el bucle de la línea 10 recorría el vector con las horas y minutos que recibía el módulo como parámetro, almacenando cada pareja en `hora_minutos`. Para cada una, rescataba la hora y

los minutos en las líneas 11 y 12, para horadar con ellas el reloj en la línea 13. No podía ser una solución más sencilla y elegante.

30.2. EL RELOJ DE SOL CONTINUO

—Ahora debemos crear el otro reloj —Antonia también parecía querer terminar hoy.

```
1 module reloj_de_sol_continuo(){
2     difference(){
3         cuerpo(largo_reloj);
4         // HACER: horas, minutos y separador
5     }
6 }
```

»Tanto las decenas como las unidades de las horas y los minutos, así como el separador, deberán recibir un tratamiento propio y particular —volvió a recordar Antonia—. Empecemos por las unidades de los minutos, si te parece —y sin más ceremonia, acercó el teclado a Cecilia.

LAS UNIDADES DE LOS MINUTOS

Cecilia se asombró al descubrir que, a pesar de no saber muy bien lo que iba a escribir, lo tomaba con seguridad: ya empezaba a comprobar qué tan cierto era el hecho de que la escritura precedía a la comprensión de un problema, y no al revés. Releyó el módulo hora_solar; en particular, la sección en la que se horadaban las unidades de los minutos:

```
1 // mucho texto antes
2     translate([0,8.5*delta_y,0])
3     digito(minuto_unidades,alfa,alfa);
4 // mucho texto despues
```

La variable delta_y era necesaria para ubicar cada dígito del reloj; debía incluir su cálculo y definición en el nuevo módulo. Por

lo demás, las unidades de los minutos eran siempre un redondo cero, abarcando desde las 6:20 a las 17:40:

```
1 module reloj_de_sol_continuo(){
2     delta_y=ancho_pixel+delta_ancho;
3     difference(){
4         cuerpo(largo_reloj);
5         // unidades de minuto
6         translate([0,8.5*delta_y,0])
7             digito(0,alfa(6+20/60),alfa(17+40/60));
8         // HACER: el resto
9     }
10 }
11 reloj_de_sol_continuo();
```

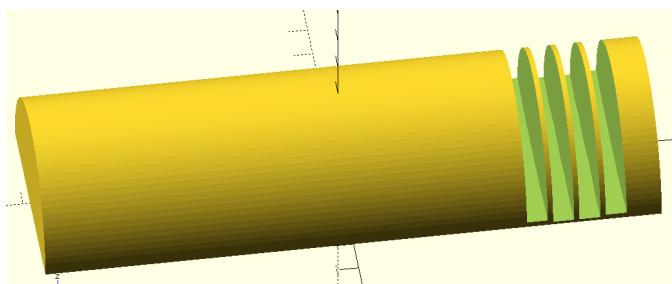


Figura 30.2: Al intentar agujerear el 0 de las unidades de los minutos, Cecilia tropieza con otro *bug*.

30.3. OTRO *bug* Y VAN...

Una sonrisa brilló rápidamente en el rostro de Cecilia: el cero apareció en su lugar, y rebanando limpiamente desde un extremo al otro el cuerpo del reloj. Sin embargo y con la misma rapidez, su sonrisa se esfumó. Pudo apreciar un claro problema: ¡El cero parecía no haber horadado el piso del reloj! Pensó que podía tratarse de un error de la vista producida por probó con .

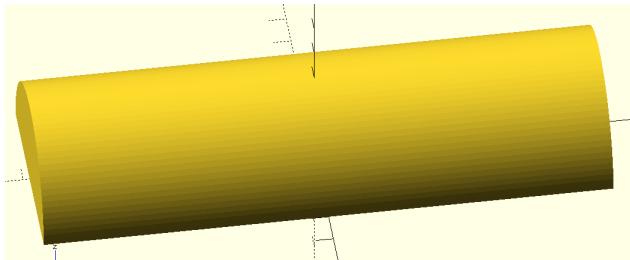


Figura 30.3: La compilación cabal del reloj no hace desaparecer el problema; antes bien, lo manifiesta con mayor crudeza.

¡Horror! ¡La figura 30.3 acusaba que ahora las mismas rebanadas desaparecían! Giró hacia Antonia en busca de ayuda; no le sorprendió demasiado encontrarla sonriendo.

—A mí me pasó exactamente lo mismo —confesó su amiga—. Es un *bug* bastante sutil; fíjate —adelantó, tomando el teclado:

```
1 echo(alfa(6+20/60));
2 echo(alfa(17+40/60));
```

```
Consola
Compiling design (CSG Tree generation)...
ECHO: 175
ECHO: 5
Compiling design (CSG Products generation)
```

Figura 30.4: Antonia señala el origen profundo del *bug*.

Cecilia miró el resultado de la consola en la figura 30.4 sin comprender nada aún.

—A las 6:20 le corresponde un ángulo α más grande que a las 17:40 —señaló Antonia.

Cecilia podía verlo con toda claridad:

—¿Y? —preguntó, sin disimular una nota de mal humor.

Antonia, ahora sin decir palabra, escribió un ejemplo más:

```
digito(0, alfa(12), alfa(13));
```

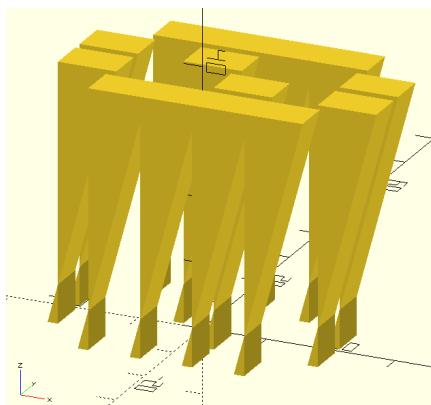


Figura 30.5: Antonia exhibe el resultado de pretender construir un rayo de Sol en el que $\alpha_1 > \alpha_2$.

Cecilia pareció querer lanzarse sobre la figura 30.5: ahora le parecía empezar a comprender el problema. Si el ángulo inicial del haz de rayos era mayor al ángulo final ($\alpha_1 > \alpha_2$), la geometría conquistada en el capítulo 29 se trastocaba, haciendo que los extremos superiores del haz resultaran “intercambiados” entre sí, con lo que el polígono presentaba una rara intersección interna.

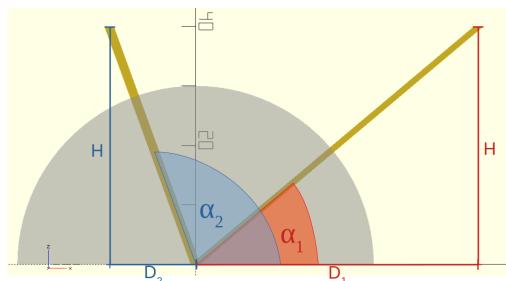
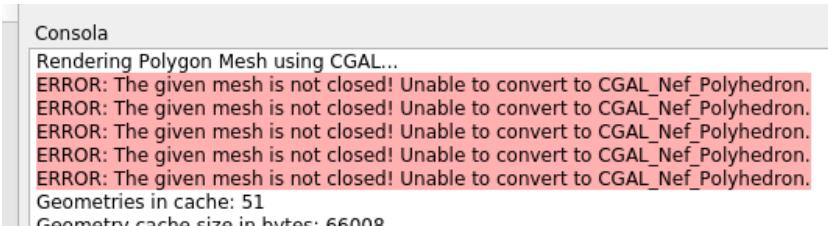


Figura 30.6: El algoritmo que Antonia y Cecilia discurrieron en el capítulo 29 para crear un haz de rayos suponía que $\alpha_1 < \alpha_2$; en caso contrario, D_1 y D_2 se trastocan y el polígono resultante no puede ser extrudido válidamente.

—Un polígono así construido, una vez extrudido, no es un objeto tridimensional legítimo —explicó Antonia—. OpenSCAD no puede determinar cuál es su interior y cuál su exterior. De hecho, si vos hacés  se va a quejar, como podés comprobar en la figura 30.7.



```

Consola
Rendering Polygon Mesh using CGAL...
ERROR: The given mesh is not closed! Unable to convert to CGAL_Nef_Polyhedron.
ERROR: The given mesh is not closed! Unable to convert to CGAL_Nef_Polyhedron.
ERROR: The given mesh is not closed! Unable to convert to CGAL_Nef_Polyhedron.
ERROR: The given mesh is not closed! Unable to convert to CGAL_Nef_Polyhedron.
ERROR: The given mesh is not closed! Unable to convert to CGAL_Nef_Polyhedron.
Geometries in cache: 51
Geometry cache size in bytes: 66008

```

Figura 30.7: La consola acusa la invalidez del haz de rayos construido con $\alpha_1 > \alpha_2$.

—¡Pues bien! Entonces la solución es fácil —dijo Cecilia, recuperando el teclado con entusiasmo.

```

1 module reloj_de_sol_continuo(){
2   delta_y=ancho_pixel+delta_ancho;
3   difference(){
4     cuerpo(largo_reloj);
5     // unidades de minuto
6     translate([0,8.5*delta_y,0])
7     digito(0,alfa(17+40/60),alfa(6+20/60));
8     // HACER: el resto
9   }
10 }
11 reloj_de_sol_continuo();

```

Antonia, aun ante la figura 30.8, no parecía satisfecha:

—El problema es que ahora va a funcionar para lectores del hemisferio sur; para los boreales, a las 17:40 les corresponde un ángulo mayor que a las 6:20 —dijo, recuperando el teclado.

```
1 hemisferio="norte";
```

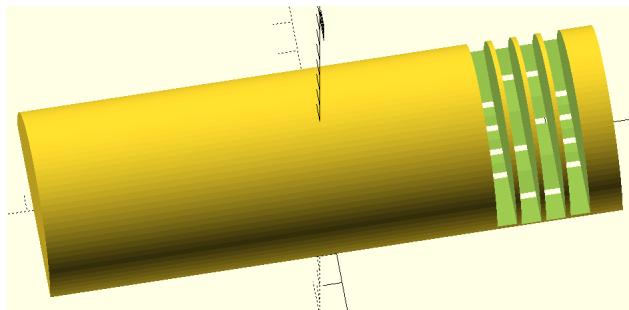


Figura 30.8: Cecilia soluciona el *bug*... aparentemente.

```
2 echo(alfa(6+20/60));  
3 echo(alfa(17+40/60));
```



Figura 30.9: Antonia demuestra que la solución rápidamente ofrecida por Cecilia no es tal.

Cecilia se derrumbó contra el respaldo de la silla; volvió a sentir que el reloj de Sol se alejaba una vez más. ¿Podrían terminarlo hoy? ¿Podrían terminarlo alguna vez? Se sentía al borde del desánimo, pero se obligó a pensar una solución; tras unos instantes, aventuró:

—¿Lo arreglamos con un *if*?

—Podría ser, pero antes debemos decidir dónde lo hacemos —aconsejó Antonia—. En este módulo me parece mala idea: no es su responsabilidad traficar con ángulos, sino con horas: ¿qué pueden importarle esos detalles de implementación? Además, imaginate que en un futuro, releyendo el texto, decidimos que es mejor medir los ángulos desde el mediodía en lugar de hacerlo desde el

horizonte: tendríamos que modificar este módulo también.

A Cecilia ya le estaba costando imaginarse terminando el reloj; no hablemos ya de releerlo en un futuro. Pero eligió seguir acompañando a su amiga.

—`reloj_de_sol_continuo` invoca al módulo `digito`, el cual sólo recibe los ángulos extremos para pasárselos a `haz_de_sol` —recapituló Antonia—. Es este último módulo el que debe asegurarse de que α_1 sea menor que α_2 —aseguró.

30.4. FUNCIONES MIN Y MAX

—Podríamos usar un `if`, o incluso el operador ternario ‘? :’ —concedió Antonia—. Pero creo que es más piola aprovechar las funciones `min` y `max`: la primera te devuelve el menor de los valores que le pasás como argumentos, y la segunda... bueno, ya lo habrás imaginado:

```
1 echo(min(5,1,2,10,3));
2 echo(max(5,1,2,10,3));
```

```
Consola
Compiling design (CSG Tree generation)...
ECHO: 1
ECHO: 10
Compiling design (CSG Product generation)
```

Figura 30.10: Antonia muestra el empleo de las funciones `min` y `max`.

»Con esas funciones, la modificación es relativamente trivial: —afirmó Antonia.

```
1 module haz_de_sol(alfa1,alfa2){
2   alfa_min=min(alfa1,alfa2);
3   alfa_max=max(alfa1,alfa2);
4   D1=H/tan(alfa_min);
5   D2=H/tan(alfa_max);
```

```
6     vertices=[[-alto_pixel/2,0] ,  
7                 [D2-alto_pixel/2,H] ,  
8                 [D1+alto_pixel/2,H] ,  
9                 [alto_pixel/2,0]];  
10    // TODO: medir la duracion de esta solucion  
11    //  
12    //      y la de esta otra:  
13    //  
14    //      translate([0,ancho_pixel/2,0])  
15    //      Djo : borrar 'center=true' abajo  
16    rotate([90,0,0])  
17    linear_extrude(ancho_pixel,center=true)  
18        polygon(vertices);  
19 }
```

A Cecilia le gustó la solución: en las líneas 2 y 3 se obtenían el menor y el mayor de los valores de α_1 y α_2 , y con ellos se calculaban D_1 y D_2 en las líneas 4 y 5. Además, cualquier modificación futura referente al tratamiento de los ángulos quedaba encapsulada en este módulo, por lo que no debería ser muy difícil de llevar a cabo. Pero lo mejor de todo es que... ¡Funcionaba!

```
1 hemisferio="sur";  
2 reloj_de_sol_continuo();
```

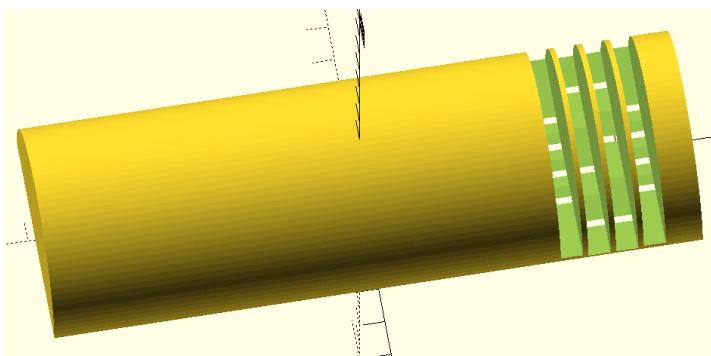


Figura 30.11: La solución de Antonia funciona para el hemisferio sur...

```

1 hemisferio="norte";
2 reloj_de_sol_continuo();

```

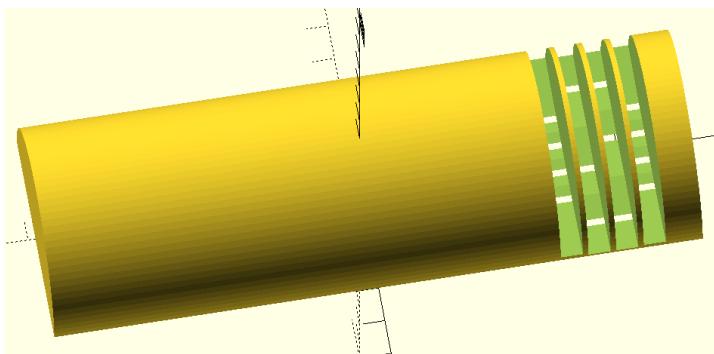


Figura 30.12: ...y para el hemisferio norte.

LAS DECENAS DE LOS MINUTOS

—Ahora es el turno de las decenas de los minutos —dijo Antonia cediendo nuevamente el teclado a Cecilia, quien lo tomó mientras releía el módulo hora_solar para inspirarse.

```

1 // [...]
2 minuto_decenas=n_a_digito(minutos,1);
3 // [...]
4 translate([0,3.5*delta_y,0])
5 digito(minuto_decenas,alfa,alfa);
6 // [...]

```

Al menos superficialmente las decenas resultaban muy parecidas a las unidades; la mayor diferencia parecía ser que el dígito no era siempre el mismo, sino que debía fluctuar entre 0, 2 y 4. Aunque examinándolo con mayor atención descubrió que el problema era más arduo de lo que parecía: ¡Los ángulos α_1 y α_2 dependían no sólo de los minutos, sino de la hora completa!

Tras unos instantes en los que se sintió prácticamente paralizada, decidió que simplemente mirando y pensando no iba a llegar a ningún lado; por lo que se lanzó a escribir, confiando en sus siempre fieles  y  Al cabo de unos intensos minutos, que para ella transcurrieron en un espacio fuera del tiempo, logró lo siguiente:

```
1 module reloj_de_sol_continuo(){
2     delta_y=ancho_pixel+delta_ancho;
3     difference(){
4         cuerpo(largo_reloj);
5         // unidades de minuto
6         translate([0,8.5*delta_y,0])
7             digito(0,alfa(6+20/60),alfa(17+40/60));
8         // decenas de minuto
9         translate([0,3.5*delta_y,0])
10        for(hora=[6:17],
11            minutos=[0,20,40])
12            if((hora+minutos/60) > 6) {
13                minuto_decenas=n_a_digito(minutos,1);
14                digito(minuto_decenas,
15                    alfa(hora+minutos/60),
16                    alfa(hora+minutos/60));
17            }
18        }
19    }
20    reloj_de_sol_continuo();
```

Una vez escrito no le pareció tan difícil... ¡Pero le costó lo suyo! Y por eso mismo estaba más contenta releyendo su nuevo texto y contemplando el resultado en la figura 30.13.

La línea 9 se encargaba de ubicar el dígito en su lugar. En las líneas 10 y 11 se declaraba el doble bucle que asignaba a hora los valores del 6 al 17 y a minutos las valores 0, 20 y 40: de esta manera se recorrían todas las horas desde las 6:00 a las 17:40 en intervalos de 20 minutos. Para evitar las conflictivas 6:00, decidió

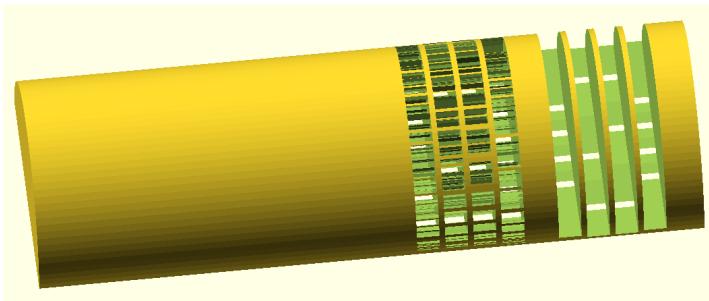


Figura 30.13: Cecilia logra horadar el reloj con las decenas de los minutos.

anteponer a la creación del dígito un `if` en la línea 12.

En la línea 13 se rescataba el dígito a horadar, y finalmente las líneas 14 a 16 se encargaban del trabajo tan delicadamente preparado por las anteriores.

Antonia sólo sonreía, feliz.

EL SEPARADOR

El separador le recordó las unidades de los minutos: un único símbolo abarcando las horas extremas.

```

1  module reloj_de_sol_continuo(){
2      delta_y=ancho_pixel+delta_ancho;
3      difference(){
4          cuerpo(largo_reloj);
5          // unidades de minuto
6          translate([0,8.5*delta_y,0])
7              digito(0,alfa(6+20/60),alfa(17+40/60));
8          // decenas de minuto
9          translate([0,3.5*delta_y,0])
10         for(hora=[6:17],
11             minutos=[0,20,40])
12             if ((hora+minutos/60)>6) {

```

```
13         minuto_decenas=n_a_digito(minutos,1);
14         digito(minuto_decenas,
15                 alfa(hora+minutos/60),
16                 alfa(hora+minutos/60));
17     }
18     // separador
19     separador(alfa(6+20/60),alfa(17+40/60));
20 }
21 }
22 reloj_de_sol_continuo();
```

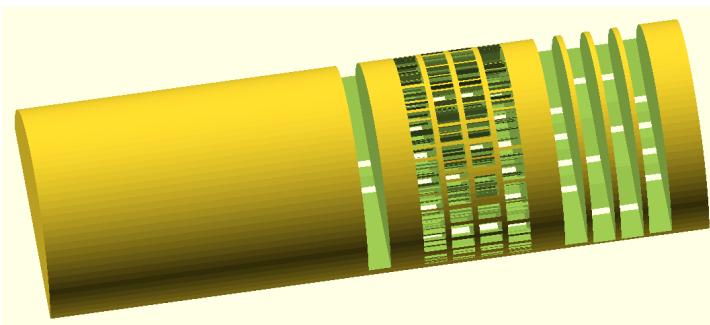


Figura 30.14: Cecilia agrega el separador de horas y minutos.

Cecilia giró para mirar a Antonia. Tal vez por primera vez desde que comenzaron esta aventura, ninguna de las dos supo qué decir.

LAS UNIDADES DE LA HORA

Sólo faltaban dos dígitos; Cecilia sentía que ya nada las podía detener. Con entusiasmo y una confianza dichosa se lanzó a escribir casi sin detenerse antes a pensar. Si la pulsación de las teclas **[Supr]** y **[←]** ocuparan sitio en el texto, este manual doblaría la cantidad de sus páginas; afortunadamente no es así.

```

1 module reloj_de_sol_continuo(){
2     delta_y=ancho_pixel+delta_ancho;
3     difference(){
4         cuerpo(largo_reloj);
5         // unidades de minuto
6         translate([0,8.5*delta_y,0])
7             digito(0,alfa(6+20/60),alfa(17+40/60));
8         // decenas de minuto
9         translate([0,3.5*delta_y,0])
10        for(hora=[6:17],
11            minutos=[0,20,40])
12            if ((hora+minutos/60)>6) {
13                minuto_decenas=n_a_digito(minutos,1);
14                digito(minuto_decenas,
15                    alfa(hora+minutos/60),
16                    alfa(hora+minutos/60));
17            }
18        // separador
19        separador(alfa(6+20/60),alfa(17+40/60));
20        // unidades de hora
21        translate([0,-3.5*delta_y,0])
22        for(hora=[6:17]){
23            hora_unidades=n_a_digito(hora,0);
24            digito(hora_unidades,
25                alfa(hora>6?hora:hora+20/60),
26                alfa(hora>6?hora:hora+20/60));
27        }
28    }
29 }
30 reloj_de_sol_continuo();

```

Cecilia saludó con una amplia sonrisa el resultado desplegado en la figura 30.15, mientras volvía a satisfacer su vanidad releyendo su flamante texto. Ahora sólo era necesario recorrer las horas mediante un bucle en la línea 22. En la 23 obtenía sus unidades. Por último, sólo restaba agujerear el reloj con el dígito correspondiente en las líneas 24 a 26.

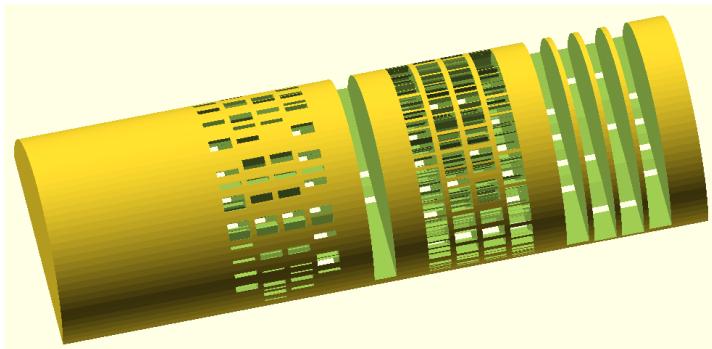


Figura 30.15: Cecilia añade al reloj las unidades de las horas.

Estaba particularmente orgullosa por la forma en que había evitado las siempre conflictivas 6:00: mediante el operador ternario '`? :'` indagaba si hora era mayor a 6. En caso afirmativo, α se calculaba simplemente usando la hora; en caso contrario, se le sumaban 20 minutos a la misma, calculando así efectivamente el ángulo para las 6:20. Se sentía muy astuta.

Cuando quiso compartir su dicha con Antonia, la encontró con un gesto ambiguo. Automáticamente se puso en guardia: se preparó para el implacable baldazo de agua fría. Sin embargo, se sorprendió al descubrir que esa actitud instintiva no era acompañada de una sensación de abatimiento: de alguna manera había logrado confiar en que podría resolver los problemas que se interpusieran entre ella y el reloj de Sol digital. Este pensamiento la hizo más feliz, quizás, que el hecho de conseguirlo en sí mismo.

—¿Qué hay ahora, Antonia? ¿Qué debemos solucionar? —preguntó Cecilia, con una sonrisa confiada.

Antonia pareció adivinar sus sensaciones, porque le devolvió una sonrisa en la que brillaba una luz que podría decirse de orgullo ante la saludable evolución operada en Cecilia, que —como casi cualquier docente— creía ingenuamente poder atribuir a

su eficacia didáctica:

—Acabás de agujerear el reloj con horas *justas*. Tal vez *demasiado* justas —observó, enigmática—. Las 12:00 te quedan perfectas, pero... ¿Qué observaremos a las, digamos, 12:20?

```
1 $vpr=[90-alfa(12),0,90];
2 reloj_de_sol_continuo();
```

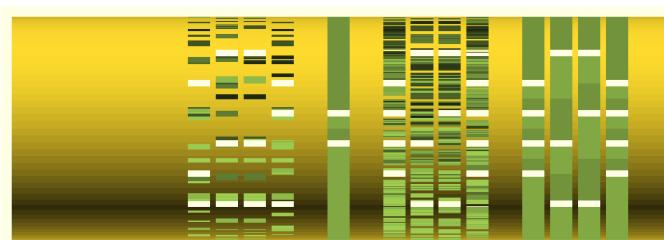


Figura 30.16: Antonia muestra que las 12:00 se observan perfectamente...

```
1 $vpr=[90-alfa(12+20/60),0,90];
2 reloj_de_sol_continuo();
```

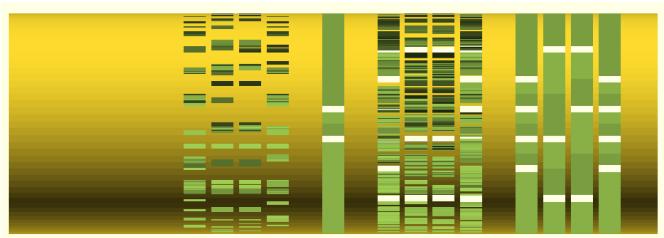


Figura 30.17: ...pero las que 12:20 no, debido a que Cecilia horadó las horas con *demasiada* justeza.

—¡Es verdad! —exclamó Cecilia, con sorpresa pero sin temor frente a las correspondientes figuras 30.16 y 30.17—. Los haces

son demasiado estrechos: deben abarcar una hora, después de todo —y con entusiasmo se dispuso a enmendar el texto.

```
1 // [...]
2 // unidades de hora
3 translate([0, -3.5*delta_y, 0])
4   for(hora=[6:17]){
5     hora_unidades=n_a_dígito(hora,0);
6     dígito(hora_unidades,
7             alfa(hora>6?hora:hora+20/60),
8             alfa(hora+59/60));
9   }
10 // [...]
11 $vpr=[90-alfa(12+20/60),0,90];
12 reloj_de_sol_continuo();
```

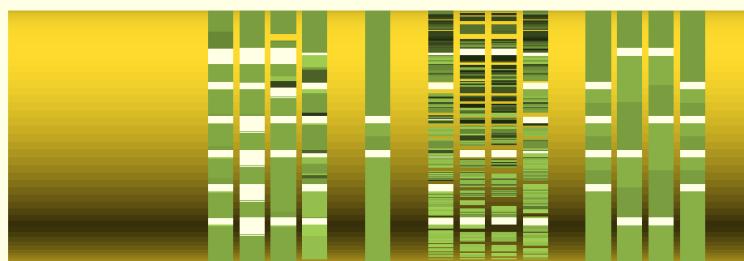


Figura 30.18: Cecilia extiende al haz de las horas 59 minutos...

El resultado exhibido en la figura 30.18 no era francamente el que Cecilia esperaba, y eso que sólo había modificado la línea 8, extendiendo el haz de cada hora hasta el minuto 59. Rotó un poco el reloj para ver si obtenía una pista de lo que estaba ocurriendo.

—¡Claro! —dijo, mientras reía y se cubría los ojos con una mano, tras enfrentarse al resultado expuesto en la figura 30.19—. ¡Ahora el haz es demasiado extenso! En esencia recaí en el error que habíamos cometido en el capítulo 27 al trazar *todas* las horas y los minutos: rebanamos el cuerpo entero del reloj.

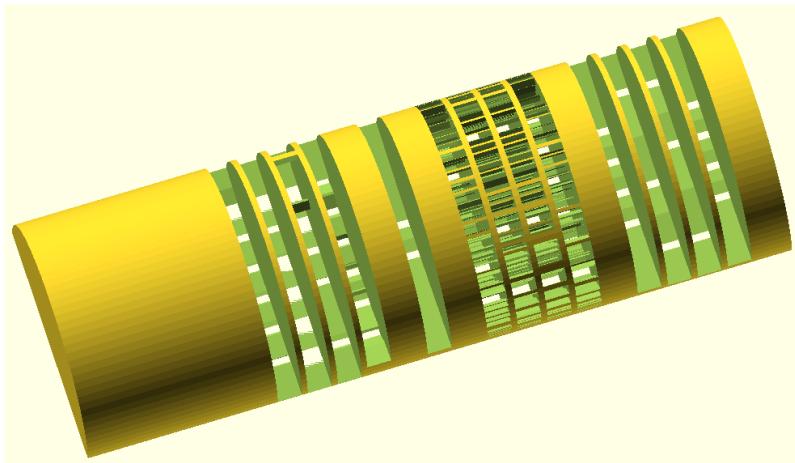


Figura 30.19: ...pero eso, ahora, parece excesivo.

—Exacto —confirmó Antonia—. Hay que usar un haz más amplio, pero no tanto como para que una hora se solape con la siguiente.

Cecilia expresó su acuerdo con un lento movimiento de cabeza:
—¿Y qué tan amplio debe ser? —indagó.

Antonia suspiró con fuerza, mientras se recostaba contra el respaldo de la silla:

—Te confieso que ésta es la parte que más me costó definir en la versión del reloj que escribí hace tiempo... Lo mejor es ir probando —sugirió.

Cecilia lo hizo con varios valores en la línea 8, pero ninguno parecía funcionar: o la hora en cuestión se “apagaba” antes de tiempo —si el haz era demasiado estrecho— o se mezclaba con una contigua —si era demasiado amplio.

Cuando estaba empezando a cansarse de probar, Antonia por fin se decidió a intervenir:

—Esas horas son un problema, ciertamente —confirmó—.

Cuando llegué a este punto me decidí a mirar el texto del creador del reloj original.¹ Tomó un par de decisiones que llamaron mi atención: la altura de cada pixel era de apenas 0,75 mm, y las horas agujereadas iban desde las 10:00 a las 16:00.

Cecilia abrió los ojos con asombro:

—¿Tan pocas horas? —preguntó.

—Sí —respondió Antonia, encogiéndose de hombros—. Tal vez es la única manera que encontró de resolver esta cuestión.

Cecilia decidió darle una oportunidad al tamaño del pixel, pero le parecía una pena renunciar a tantas horas del día. Sin embargo, y después de mucho probar, lo mejor que consiguió fue desplazar una hora el mismo conjunto de 6 horas, abarcando así un intervalo más simétrico desde las 9:00 a las 15:00:

```
1 alto_pixel = .75; // Era 1.6 y antes 2
2 // [...]
3 // unidades de hora
4 translate([0, -3.5*delta_y, 0])
5 for(hora=[9:15]){
6     hora_unidades=n_a_digito(hora,0);
7     digito(hora_unidades,
8             alfa(hora),
9             alfa(hora<15 ? hora+50/60:
10                hora+10/60));
11 }
12 // [...]
13 $vpr=[90-alfa(12+40/60),0,90];
14 reloj_de_sol_continuo();
```

En la línea 5 el bucle se resignaba a las horas que iban desde las 9 a las 15. Cada una de ellas se extendía durante 50 minutos en la línea 9 (salvo las 15:00, que se prolongaban sólo 10 minutos), por lo que el “día” del reloj acababa reduciéndose al intervalo que

¹No está de más recordar aquí el origen de la inspiración del reloj: <https://www.thingiverse.com/thing:1068443>. (Nota del Editor)

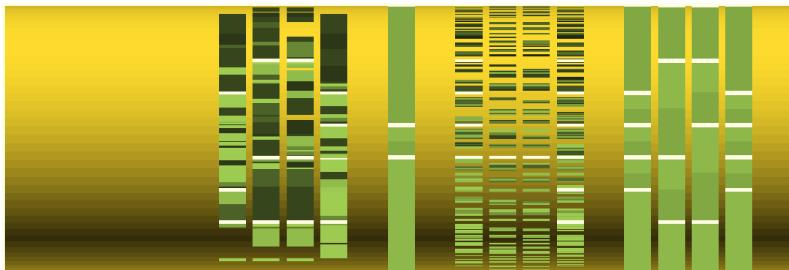


Figura 30.20: Cecilia se resigna a horadar el reloj con las horas comprendidas entre las 9:00 y las 15:00. Además, usa pixeles menos altos.

iba desde las 9:00 a las 15:10.

Cecilia sentía una mezcla agridulce de sensaciones, en las que se entrelazaban la nostalgia por haber perdido la totalidad de las horas diurnas y la alegría de estar a un dígito de terminar su ansiado reloj. Mientras releía su texto reconoció que en última instancia primaba una suerte de dicha: reconoció en esa mezcla indecisa y anhelante un efecto muy parecido al que le producían la música, la literatura y el arte en general; esa inquietante impresión que describió Edgar Allan Poe con una admirable queja: el Arte nos depara la felicidad y la congoja de quien ve los Cielos súbitamente abiertos para inmediatamente después encontrarlos cerrados y ocultos tras un denso y opaco velo.

Antonia la devolvió bruscamente a la realidad:

—¿Qué hacemos ahora con los minutos? —preguntó.

Cecilia emergió de sus pensamientos con cierta alarma:

—¿Cómo qué hacemos? Ya está; ya los hicimos —se defendió, temiendo lo peor.

Antonia sonrió, divertida:

—Me parece que no tanto; en principio, si las horas van de las 9:00 a las 15:10 no tiene sentido que los minutos comiencen antes

y terminen después...

Cecilia festejó la observación con una risa nerviosa:

—¡Me asustaste, Antonia! Pensé que era algo más grave.

—Es que hay otro problema —agregó su amiga, tomando el teclado para sí—; fíjate que pasa a las, digamos... 12:50.

```
1 $vpr=[90-alfa(12+50/60),0,90];  
2 reloj_de_sol_continuo();
```

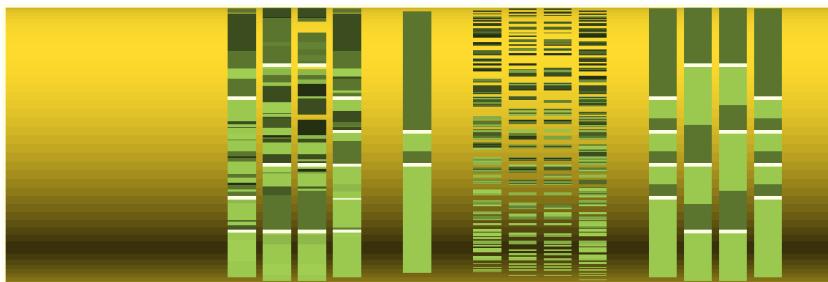


Figura 30.21: Antonia muestra que los haces de los minutos, también, deben extenderse un tanto en el tiempo.

—Nooo... —gimió Cecilia. Y tras unos instantes, añadió—: Claro, ahora que los pixeles son menos altos, los haces de un solo rayo resultan prácticamente ‘instantáneos’. En fin, no parece tan difícil de solucionar —y recuperando el teclado se dispuso a enmendar ambos problemas con renovado optimismo.

```
1 module reloj_de_sol_continuo(){  
2     delta_y=ancho_pixel+delta_ancho;  
3     difference(){  
4         cuerpo(largo_reloj);  
5         // unidades de minuto  
6         translate([0,8.5*delta_y,0])  
7             digito(0,alfa(9),alfa(15+10/60));  
8             // decenas de minuto  
9             translate([0,3.5*delta_y,0]) {
```

```

10     for(hora=[9:14] ,
11         minutos=[0,20,40]){
12         minuto_decenas=n_a_digito(minutos,1);
13         digito(minuto_decenas,
14                 alfa(hora+minutos/60),
15                 alfa(hora+(minutos+10)/60));
16     }
17     digito(0,alfa(15),alfa(15+10/60));
18 }
19 // separador
20 separador(alfa(9),alfa(15+10/60));
21 // unidades de hora
22 translate([0,-3.5*delta_y,0])
23     for(hora=[9:15]){
24         hora_unidades=n_a_digito(hora,0);
25         digito(hora_unidades,
26                 alfa(hora),
27                 alfa(hora<15 ? hora+50/60 :
28                     hora+10/60));
29     }
30 }
31 $vpr=[90-alfa(12+50/60),0,90];
32 reloj_de_sol_continuo();

```

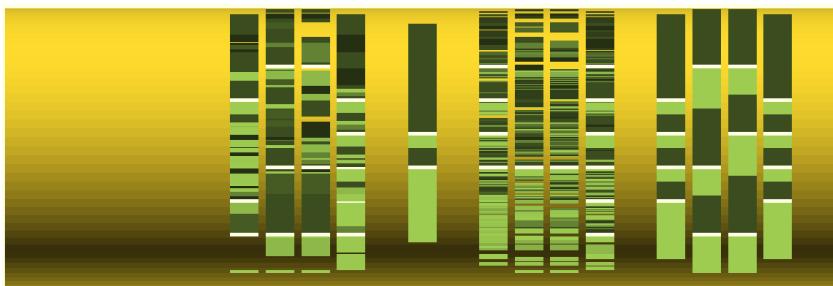


Figura 30.22: Cecilia extiende la duración de los haces de las decenas de los minutos.

Lo más fácil fue coordinar las horas de todos los elementos: bastó con modificar las líneas 7 (para las unidades de los minutos), la 10 (para sus decenas) y la 20 (para el separador). Quitó también el `if` que precedía a la creación de las decenas de los minutos (ya que las 6:00 quedaban automáticamente excluidas por la misma definición del bucle correspondiente), pero agregó en la línea 17 el caso especial del '0' de las 15:00, puesto que resultaba más difícil incluirlo en dicho bucle.

Por otro lado, para que las decenas de los minutos duraran lo más posible extendió 10 minutos sus respectivos haces en la línea 15. Lo que más le costó fue decidirse por ese valor en particular; probó varios intervalos, aunque ninguno la satisfizo del todo: si el intervalo era muy breve los dígitos se apagaban muy pronto, y si era muy grande algunos píxeles se colaban en dígitos vecinos. Esto último, de todas maneras, ocurría incluso en algunas horas aunque en menor medida empleando los 10 minutos elegidos: supuso, quizá con más cansancio que convicción, que con tantos haces de luz atravesando el reloj tal comportamiento sería inevitable.²

¹ `$vpr=[90 - alfa(12+30/60) ,0 ,90];`

² `reloj_de_sol_continuo();`

—Las 12:20, que deben mostrarse a las 12:30, quedan un poco ‘sucias’ con píxeles del ‘4’ vecino encendidos —se lamentó Cecilia en voz alta, requiriendo así la opinión de su amiga, o tal vez su consejo.

—A mí me pasó lo mismo —fue la lacónica respuesta de Antonia.

² Quedan como ejercicios para el lector la busca de otros valores para las variables del reloj que eviten ésta y cualquier otra imperfección, así como el logro de un mundo perfecto.

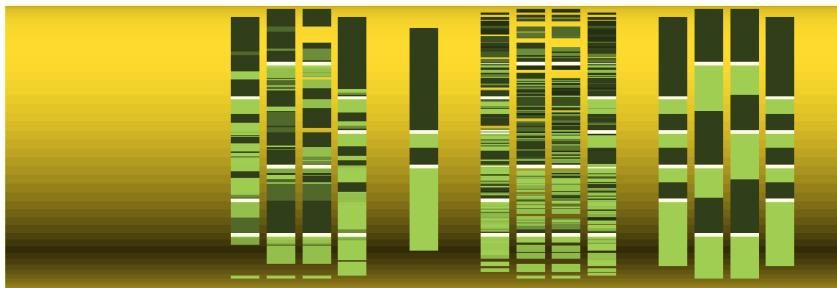


Figura 30.23: Cecilia comprueba con resignación y melancolía que su solución para la duración de las decenas de los minutos no es tan ideal como soñaba.

LAS DECENAS DE LA HORA

Lo que restaba parecía tan fácil como las unidades de los minutos: tan solo un '1' constante.

```

1 module reloj_de_sol_continuo(){
2     delta_y=ancho_pixel+delta_ancho;
3     difference(){
4         cuerpo(largo_reloj);
5         // unidades de minuto
6         translate([0,8.5*delta_y,0])
7             digito(0,alfa(9),alfa(15+10/60));
8         // decenas de minuto
9         translate([0,3.5*delta_y,0]){
10             for(hora=[9:14],
11                 minutos=[0,20,40]){
12                 minuto_decenas=n_a_digito(minutos,1);
13                 digito(minuto_decenas,
14                     alfa(hora+minutos/60),
15                     alfa(hora+(minutos+10)/60));
16             }
17             digito(0,alfa(15),alfa(15+10/60));
18         }
}

```

```

19    // separador
20    separador(alfa(9), alfa(15+10/60));
21    // unidades de hora
22    translate([0, -3.5*delta_y, 0])
23        for(hora=[9:15]){
24            hora_unidades=n_a_digito(hora,0);
25            digito(hora_unidades,
26                    alfa(hora),
27                    alfa(hora<15 ? hora+50/60 :
28                        hora+10/60));
29        }
30    // decenas de hora
31    translate([0, -8.5*delta_y, 0])
32        digito(1, alfa(10), alfa(15+10/60));
33    }
34 $vpr=[90-alfa(12+40/60), 0, 90];
35 reloj_de_sol_continuo();

```

Cecilia casi no se sorprendió al comprobar que bastaban sólo dos líneas: la 30 —que ubicaba el ‘1’ en su debido lugar— y la 31 —que lo desplegaba desde las 10 hasta las 15:10.

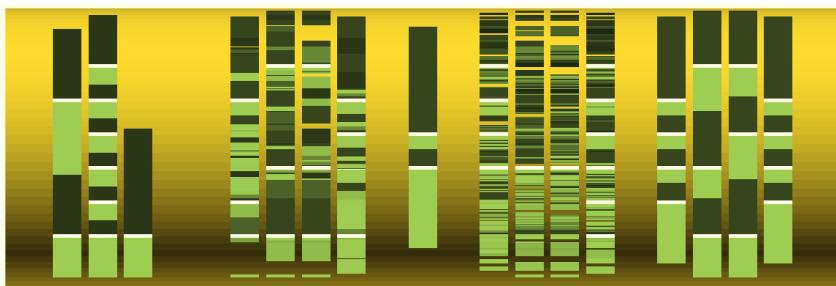


Figura 30.24: Cecilia horada el reloj con el 1 de las decenas de las horas.

Cecilia se recostó blandamente contra el respaldo de la silla; dirigía su mirada alternativamente al monitor y a Antonia. Quizás

esperaba de ella alguna objeción, u observación que implicara una nueva dilación en la conquista del reloj. Pero su amiga sólo le devolvía la mirada, sonriendo.

—¿Y? —se animó por fin a preguntar—. ¿Ya está?

—Yo diría que sí. Ya está —respondió Antonia.

—Así que ya está —fue lo único que Cecilia pudo deslizar, en un susurro.

Ya estaba.

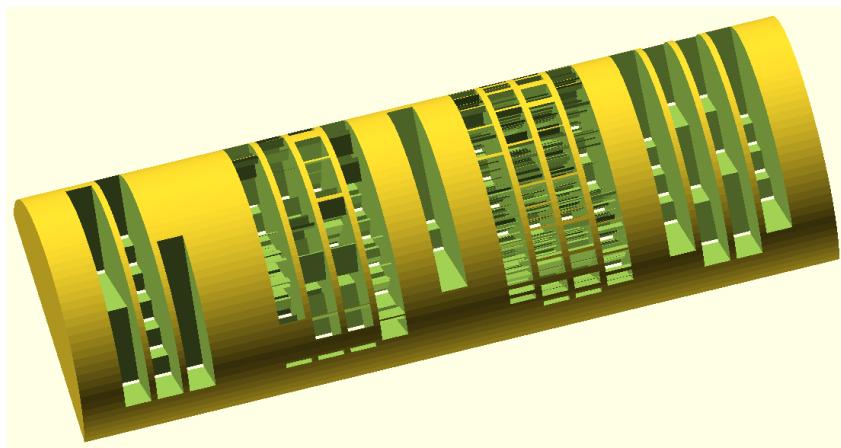


Figura 30.25: Cecilia y Antonia logran finalmente su reloj de Sol digital.

Las últimas pinceladas

AL DÍA SIGUIENTE Cecilia no entró sola a la oficina de Antonia: la acompañaba una fragante docena de medialunas. Su amiga, por primera vez en mucho tiempo, despejó de papeles el escritorio: tal compañía merecía un lugar especial.

El Sol asomaba sus rayos por la amplia ventana que daba a los jardines, sumando su calidez al ambiente festivo que producían las risas contagiosas de ambas amigas. Reían con la facilidad que permite la amistad, sin que lo exija un chiste u observación demasiado ocurrente. Simplemente se sentían dichosas.

Cuando la docena se transformó en seis, Antonia se limpió un poco los dedos en uno de los papeles periféricos del escritorio y se dispuso a proponer los últimos retoques al reloj de Sol digital:

—Quedamos en ofrecer al usuario la posibilidad de crear dos tipos de relojes: uno que sólo muestre unas cuantas horas puntuales escogidas por él, y otro que presente todas las que abarcan desde las 9:00 a las 15:10 —recordó.

Cecilia, recostada contra el respaldo de la silla, saboreó ese recuerdo casi con la misma fruición con la que terminaba su tercera medialuna.

—¿No sería más lindo ofrecerle un solo módulo, y que éste se encargue de hacer un reloj u otro? —preguntó Antonia con tono

evidentemente retórico.

—¿Vos decís que escribamos un módulo que lea la mente?

—Cecilia seguía de excelente humor, y sólo parecía querer tener otra excusa para reír.

—No estaría mal —respondió Antonia con una sonrisa—. Pero yo estaba pensando en algo más modesto: crear un módulo `reloj_de_sol` que examine los parámetros que recibe: en el caso de que no reciba ninguno, que cree un `reloj_de_sol_continuo`; y si recibe un vector, que produzca un `reloj_de_sol_discreto`.

—Suena bien —asintió Cecilia.

31.1. PARÁMETROS INDEFINIDOS

Antonia no había esperado la aprobación de su amiga para empezar a escribir.

```
1 /* Crea un reloj de Sol en función
2   del valor de 'vector_horas'.
3   Ver comentarios en los modulos
4   'reloj_de_sol_continuo' y
5   'reloj_de_sol_discreto'.
6 */
7 module reloj_de_sol(vector_horas){
8   if(vector_horas==undef)
9     reloj_de_sol_continuo();
10  else
11    reloj_de_sol_discreto(vector_horas);
12 }
```

—Creo que el texto se explica solo —el optimismo de Antonia seguía intacto—. En la línea 8 se indaga si el usuario invocó el módulo con un parámetro o no: la idea es que un parámetro no escrito toma un valor igual a '`undef`'. De esta forma, si no recibe ningún vector crea un reloj de Sol continuo; en caso contrario, un reloj de Sol discreto con las horas que efectivamente recibió.

```
reloj_de_sol();
```

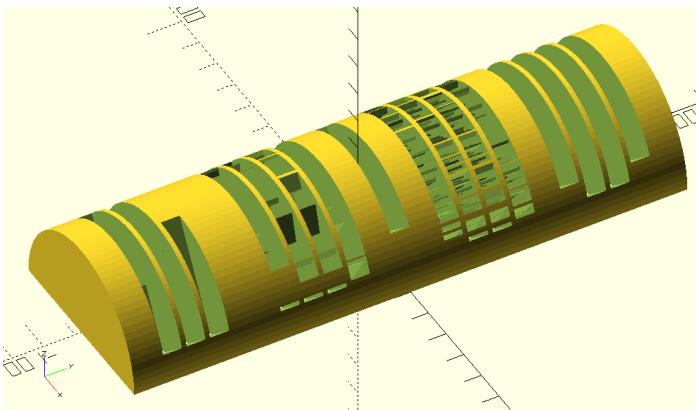


Figura 31.1: La invocación del módulo `reloj_de_sol` sin parámetros crea un reloj de Sol continuo...

```
reloj_de_sol([[12,00],[7,13],[16,23]]);
```

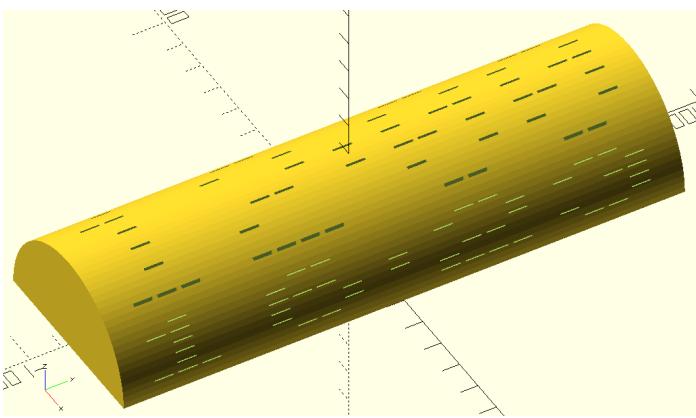


Figura 31.2: ...mientras que si se le ofrece un vector, produce un reloj de Sol discreto.

—Está bueno —aprobó Cecilia, mientras comenzaba a hacerse cargo de su cuarta medialuna.

31.2. UNA VIEJA DUDA

Antonia tomó la suya con la punta de sus dedos, para mancharlos lo menos posible y así poder seguir escribiendo:

—No habrás olvidado tampoco una antigua duda que nos acompaña desde el capítulo 19; se trataba del método óptimo para ubicar correctamente el rayo de Sol, ahora mejor conocido como ‘haz de Sol’:

```
1 module haz_de_sol(alfa1, alfa2){  
2     alfa_min=min(alfa1, alfa2);  
3     alfa_max=max(alfa1, alfa2);  
4     D1=H/tan(alfa_min);  
5     D2=H/tan(alfa_max);  
6     vertices=[[-alto_pixel/2,0],  
7                 [D2-alto_pixel/2,H],  
8                 [D1+alto_pixel/2,H],  
9                 [alto_pixel/2,0]];  
10    // TODO: medir la duracion de esta solucion  
11    //         y la de esta otra:  
12    //         translate([0, ancho_pixel/2,0])  
13    //         Ojo : borrar 'center=true' abajo  
14    rotate ([90,0,0])  
15    linear_extrude(ancho_pixel, center=true)  
16    polygon(vertices);  
17 }
```

Cecilia apenas se acordaba, pero se sintió obligada a asentir con entusiasmo:

—Dale, compilemos ambas versiones y veamos.

Con la versión actual del texto el reloj completo demoraba un minuto y medio en compilarse, mientras que la segunda versión tardaba... virtualmente lo mismo:

```
CGAL Polyhedrons in Cache: 42
CGAL cache size in bytes: 23225920
Total rendering time: 0:01:31.041
Top level object is a 3D object:
Simple: vec
```

Figura 31.3: Antonia comprueba el tiempo que demora la compilación del reloj...

```
CGAL Polyhedrons in Cache: 42
CGAL cache size in bytes: 23225920
Total rendering time: 0:01:31.101
Top level object is a 3D object:
Simple: vec
```

Figura 31.4: ...empleando las dos soluciones propuestas en el módulo `haz_de_sol`.

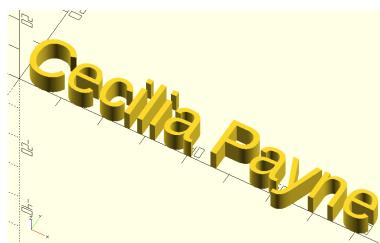
—La diferencia es ridícula; yo diría que elijamos la que nos guste más —propuso Antonia con despreocupación, dejando evidentemente la decisión en manos de su amiga.

—Me gusta más la primera —afirmó Cecilia, dando fin a su cuarta medialuna.

31.3. TEXTO

—Si no me equivoco, me falta cumplir una última promesa: la escritura de texto en OpenSCAD —dijo Antonia, mientras se apresuraba a rescatar su quinta medialuna.

```
1 linear_extrude(5)
2   text("Cecilia Payne");
```



»La forma básica no puede ser más directa y sencilla: consiste en usar la sentencia `text` con el texto deseado como parámetro —señaló Antonia—. Esta sentencia, de por sí, crea un objeto bidimensional, como lo hacen `square`, `circle` y `polygon`; por eso es necesario otorgarle tridimensionalidad mediante una transformación como `linear_extrude`.

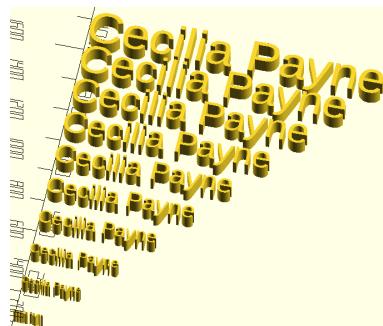
Tras una pausa para masticar, Antonia continuó:

—`text` admite varios parámetros más. Te sugiero consultar el ayudamemoria de OpenSCAD¹ para verlos todos; mientras tanto, te comento algunos.

SIZE

»Con `size` podés indicar el tamaño de las letras:

```
1 for (s=[10:10:100])
2     translate ([0,10*s,0])
3         linear_extrude (50)
4             text ("Cecilia
5                 Payne", size=s);
```



VALIGN

»Con `valign` es posible decidir la ubicación de las letras respecto de la línea base; el valor por defecto es `baseline`.

```
text ("Payne", valign="baseline");
```



¹<https://openscad.org/cheatsheet/>

```
text ("Payne", valign="bottom");
```



```
text ("Payne", valign="top");
```



```
text ("Payne", valign="center");
```



HALIGN

»También es posible precisar la posición respecto de la vertical; el valor por defecto es `left`.

```
text ("Payne", halign="left");
```



```
text ("Payne", halign="center");
```



```
text ("Payne", halign="right");
```



FONT

»Y, por supuesto, también podés elegir el tipo de letra: tanto su familia como su estilo. Para eso primero tenés que ir al menú Ayuda → Font List, como podés apreciar en la figura 31.5.

»Te aparecerá una larga lista de las fuentes presentes en la computadora, igual o parecida a la de la figura 31.6; tenés que elegir una y hacer *click* en el botón 'Copiar al portapapeles'.

»Ahora ya podés pegar el contenido del portapapeles, entre comillas, luego del parámetro `font`.

31. LAS ÚLTIMAS PINCELADAS

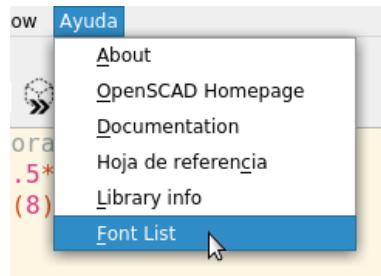


Figura 31.5: Menú Ayuda → font List.

A screenshot of the 'Lista de fuentes OpenSCAD' (OpenSCAD Font List) dialog box. The table lists various fonts installed on the system. The columns are 'Font name', 'Font style', and 'Filename'. The 'URW Chancery L' font is selected and highlighted with a blue border. The 'Copiar al portapapeles' (Copy to clipboard) button is visible at the bottom right of the dialog.

Font name	Font style	Filename
232	URW Bookman	Demi
257	URW Bookman	Demi Italic
403	URW Bookman	Light
264	URW Bookman L	Demi Bold Italic
333	URW Bookman L	Light
347	URW Bookman L	Light Italic
376	URW Bookman L	Demi Bold
72	URW Chancery L	Medium Italic
24	URW Gothic	Book Oblique

Figura 31.6: Lista de fuentes instaladas.

```
1 linear_extrude(6)
2   text("Cecilia Payne",
3       font="URW Chancery
4       L:style=Medium
5       Italic");
```



Cecilia saludó tan bello resultado dando los últimos mordiscos

a su quinta medialuna.

31.4. LA LEYENDA DEL RELOJ

—Parece que la humanidad no considera una obra debidamente acabada si no estampa sobre ella algunas palabras: placas conmemorativas en edificios, frases motivadoras en jabones de tocador, patrocinios publicitarios en camisetas deportivas son apenas unos ejemplos. Hasta la música —que, como Borges le hizo decir a Schopenhauer, es la única de las Artes que puede prescindir del Universo— debe sobrellevar las palabras: aunque Debussy impuso que los títulos de sus preludios se imprimieran al pie de los mismos, no logró que dejáramos de recordarlos por ellos —Antonia suspiró, olvidando por un momento que quedaban sólo dos medialunas.

»Los relojes de Sol no son la excepción. Estuve buscando una frase que diera algo de complejidad a sus bordes, pero ninguna satisfizo mi vanidad: opté por escribir una propia.

Cecilia sintió que debía felicitar a su amiga:

—¿En serio? ¡Qué bueno! ¿Y cuál fue?

Antonia sonrió, debidamente halagada:

—Por supuesto, debía ser en latín: *Ab Revolutionibus Astri Girum Orbis Noscimus*.

Cecilia puso a prueba sus recuerdos colegiales de latín:

—¿Puede ser que signifique algo así como 'De las revoluciones del astro sabemos el giro del orbe'?

—A mí me gusta más 'Gracias al giro (aparente) del Sol conocemos que la Tierra rota' —corrigió Antonia, sin disimular una nota de orgullo en su voz.

—Me gusta —concedió Cecilia, mientras miraba de reojo las dos últimas medialunas.

Antonia buscó una imagen en su computadora; finalmente la

encontró:

—Quería que quedara como en la figura 31.7.

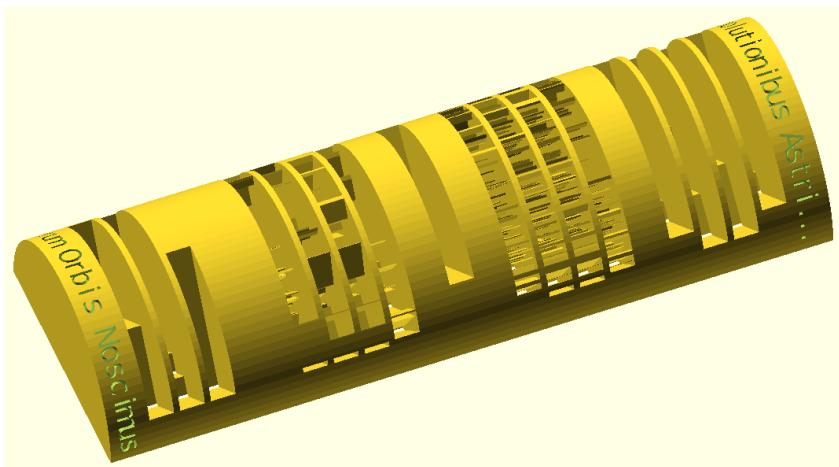


Figura 31.7: Antonia muestra cómo debería quedar la inscripción en los bordes del reloj.

»Tenemos entonces que escribir un módulo que despliegue un texto a lo largo de un bonito arco circular —agregó, mientras cedía el teclado a Cecilia.

Ésta lo tomó y, olvidando también las últimas medialunas, comenzó a pensar un voz alta:

—Yo diría que el módulo debería recibir el texto y su tamaño... también su espesor, claro. ¡Ah! Y el radio del arco circular, además del ángulo a lo largo del cual debe extenderse.

```
1 module texto_circular(texto, size, espesor,  
2   radio, angulo){  
3 }
```

31.5. LEN

«Cada letra debería rotarse un cierto ángulo a lo largo del arco, igual al ángulo total dividido por la cantidad de letras que tiene el texto» —pensó Cecilia. En seguida agregó en voz alta, casi segura de la respuesta:

—¿Hay una manera de calcular la longitud de un texto, Antonia?

Su amiga sonrió mientras recuperaba el teclado:

—Por supuesto; con la función `len`:

<pre>echo (len("Hola"));</pre>	Consola Compiling design (CSG Tree generation).. ECHO: 4 Compiling design (CSG Products generati
--------------------------------	--

¡Perfecto! —exclamó Cecilia, tomando el teclado.

```
1 module texto_circular(texto, size, espesor,
2   radio, angulo){
3   longitud=len(texto);
4   delta_alfa=angulo/longitud;
```

Antonia no esperó esta vez a que Cecilia la mirara para expresar su disconformidad:

—Hmm... no estoy de acuerdo. Los ‘huecos’ entre letras son uno menos que la cantidad de éstas; por ejemplo, si tu texto tuviera dos letras, `delta_alfa` debería ser igual a `angulo`, no a su mitad.

Cecilia admitió con una sonrisa que su amiga tenía razón.

```
1 module texto_circular(texto, size, espesor,
2   radio, angulo){
3   longitud=len(texto);
4   delta_alfa=angulo/(longitud -1);
```

«Ahora tengo que recorrer las letras del texto, una por una, para crearlas, desplazarlas y rotarlas» —pensó. Y segura de que eso tenía que ser posible, preguntó:

—¿Cómo hago para rescatar del texto una letra en particular?

—Las cadenas de texto son tratadas por OpenSCAD como vectores de caracteres, por lo que podés obtener uno cualquiera empleando un índice —explicó Antonia, mientras volvía a tomar el teclado.

```
1 texto="abcdefghijklm";
2 echo(texto[0],texto[4],texto[2]);
```

Consola
ECHO: "a", "e", "c"
Compiling design.lf

—¡Genial! —aprobó Cecilia recobrando el teclado una vez más.

```
1 module texto_circular(texto, size, espesor,
    radio, angulo){
2   longitud=len(texto);
3   delta_alfa=angulo/(longitud-1);
4   for(i=[0:longitud-1]){
5
6 }
7 }
```

—Ya es la segunda vez que aparece `longitud-1`... —protestó Antonia—. ¿Y si en lugar de almacenar la longitud del texto guardás ese otro valor aparentemente más útil?

Cecilia no encontró reparos en aceptar la sugerencia.

```
1 module texto_circular(texto, size, espesor,
    radio, angulo){
2   n=len(texto)-1;
3   delta_alfa=angulo/n;
4   for(i=[0:n]){
5
6 }
7 }
```

—De paso, tipeo menos —bromeó—. Ahora sí: a cada letra la separo del origen una distancia igual a radio, y la roto $\delta\alpha \cdot i$ grados.

```

1 module texto_circular(texto, size, espesor,
2   radio, angulo){
3   n=len(texto)-1;
4   delta_alfa=angulo/n;
5   for(i=[0:n])
6     rotate([0,0,delta_alfa*i])
7     translate([radio,0,0])
8     linear_extrude(espesor)
9     text(texto[i], size=size, font="DejaVu
  Sans:style=Book");
9 }
```

Cada letra era creada en las líneas 7 y 8, alejada del centro en la 6 y rotada en la 5.

```
texto_circular(texto="Ab Revolutionibus
Astri...", size=5, espesor=3, radio=60,
angulo=90);
```

El resultado, tal como podía apreciarse en la figura 31.8, era efectivamente un texto dispuesto circularmente, pero no como ella lo esperaba: las letras debían presentar su dorso al centro, y estar alineadas con el eje de rotación.

—Supongo que tengo que rotar las letras antes de trasladarlas —aventuró, y se dispuso a intercalar una línea más.

```

1 module texto_circular(texto, size, espesor,
2   radio, angulo){
3   n=len(texto)-1;
4   delta_alfa=angulo/n;
5   for(i=[0:n])
6     rotate([0,0,delta_alfa*i])
7     translate([radio,0,0])
8     rotate([90,0,90])
9     linear_extrude(espesor)
```

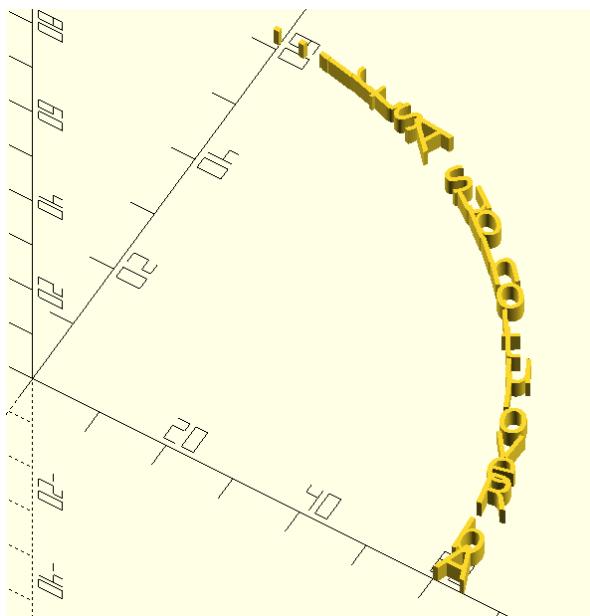


Figura 31.8: El texto es efectivamente circular, pero las letras no están dispuestas como esperamos.

```
9         text(texto[i], size=size,  
10        font="DejaVu Sans:style=Book");  
11    }
```

—¡Mucho mejor! —exclamó Cecilia admirando complacida la figura 31.9, mientras tomaba triunfalmente su sexta medialuna y la apoyaba frente a su boca como una sonrisa subrayada, antes de darle un implacable mordisco.

Antonia aplaudió con entusiasmo antes de oponer un par de objeciones y hacerse cargo de la medialuna restante:

—Fíjate que el arco comienza justo en el eje X y, pese a que indicaste un ángulo de 90° , el último punto se pasó un poquito del eje Y.

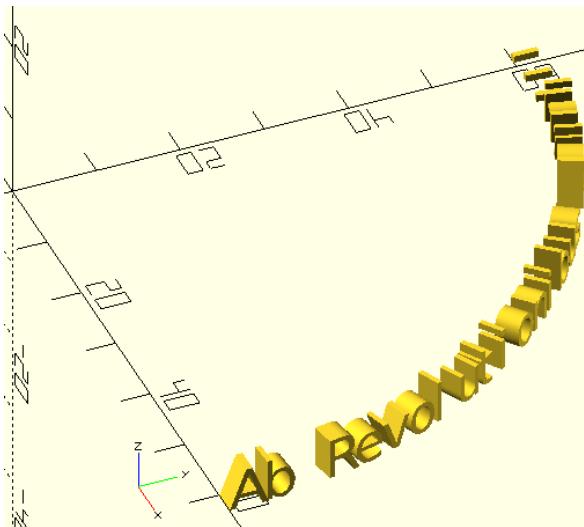


Figura 31.9: Ahora las letras presentan correctamente su dorso al eje de rotación.

Cecilia pudo verlo, pero no así el inconveniente.

—Esto pasa porque cada letra, por defecto, está alineada horizontalmente ‘por izquierda’; yo diría que resulta más natural que cada letra esté centrada —argumentó Antonia.

Cecilia ahora entendió la observación, y no le costó nada salvarla.

```

1 module texto_circular(texto, size, espesor,
2   radio, angulo){
3   n=len(texto)-1;
4   delta_alfa=angulo/n;
5   for(i=[0:n])
6     rotate([0,0,delta_alfa*i])
7     translate([radio,0,0])
8     rotate([90,0,90])
9     linear_extrude(espesor)

```

```
9     text(texto[i], size=size,
10        font="DejaVu Sans:style=Book",
11        halign="center");
12 }
```

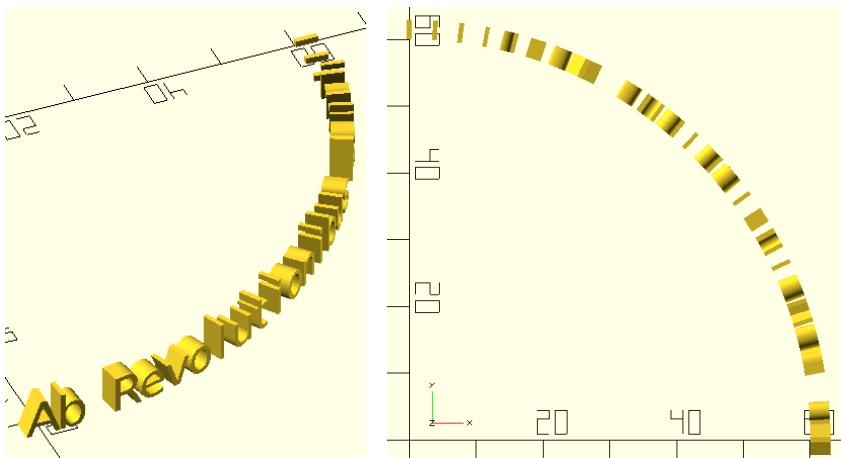


Figura 31.10: Las letras ahora están alineadas horizontalmente de manera correcta.

—Fue tan fácil como agregar un `halign="center"` en la línea 9 —comentó complacida, enseñando a su amiga la figura 31.10.

—Una última cuestión —prometió Antonia, señalando a su vez la figura 31.11—. Nosotras queremos que el texto quede centrado respecto del eje longitudinal del reloj; es decir, queremos que resulte simétrico respecto del plano YZ. Yo diría que para lograr eso conviene que el módulo `texto_circular` produzca un texto simétrico respecto del plano XZ; en otras palabras, que no empiece en el eje X, sino un ángulo igual a `angulo/2` antes —explicó Antonia.

A Cecilia le pareció razonable.

```
1 module texto_circular(texto, size, espesor,
2   radio, angulo){
```

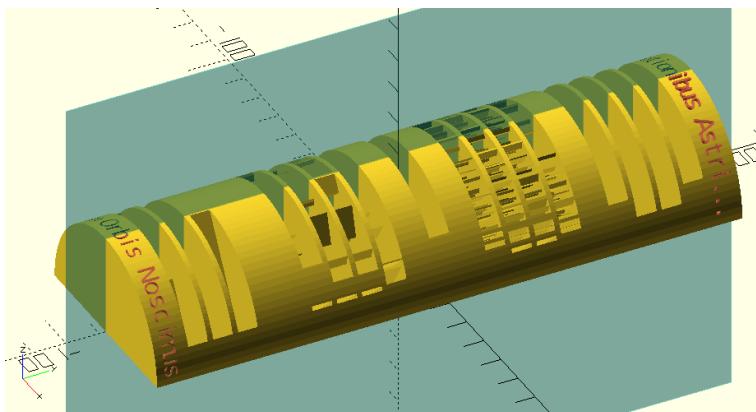


Figura 31.11: El texto debería resultar simétrico respecto del plano YZ

```

2 n=len(texto)-1;
3 delta_alfa=angulo/n;
4 for(i=[0:n])
5     rotate([0,0,delta_alfa*i-angulo/2])
6     translate([radio,0,0])
7     rotate([90,0,90])
8     linear_extrude(espesor)
9     text(texto[i], size=size,
10         font="DejaVu Sans:style=Book",
11         halign="center");
12 }
```

Sólo fue necesario modificar la línea 5: todas las letras recibían una rotación extra de $-\text{angulo}/2$ grados.

```
texto_circular(texto="Ab Revolutionibus
Astri...", size=5, espesor=3, radio=60,
angulo=90);
```

—¡Excelente! —esta vez Antonia, contemplando la figura 31.12, no tuvo nada que objetar.



Figura 31.12: El texto producido por el módulo `texto_circular` resulta ahora simétrico al plano XZ.

—Creo que el texto debería poder ser elegido por el usuario —opinó Cecilia—. Vamos a guardarlos en dos variables, junto con las demás.

```
1 texto_1 = "Ab Revolutionibus Astri...";  
2 texto_2 = "...Girum Orbis Noscimus";
```

Cecilia no estaba tan segura de que en latín todas las palabras llevaran mayúscula inicial, pero no se sentía tan solvente en gramática clásica como para contradecir a su amiga.

31.6. IMPORT

—Ahora debemos aplicar ambos textos circulares al cuerpo del reloj. El problema es que para ver cómo va quedando deberíamos compilar el reloj a cada paso —Antonia no parecía querer someter a su computadora a los mismos cálculos una y otra vez—. Así que estaría bueno poder guardar el reloj como un objeto compilado

aparte, y usarlo luego como si fuera un cuerpo básico más —y tomó el teclado para sí, dispuesta a volver ese deseo realidad.

»Una vez que creamos el reloj con  podemos grabarlo en la computadora con  o usando el ícono correspondiente de la barra —mostró Antonia, señalando la figura 31.13.

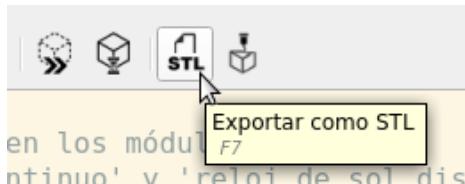


Figura 31.13: Ícono para exportar un objeto en formato STL.

»De paso teuento que el formato .STL es nada menos que el que debemos producir en última instancia para procesarlo luego mediante un programa que lo traduzca al lenguaje que las impresoras 3D entienden —explicó—. Pero ahora nos interesa porque este tipo de archivos podemos incluirlo en nuestras escenas usando la sentencia `import`.

```
import("reloj-de-sol.stl");
```

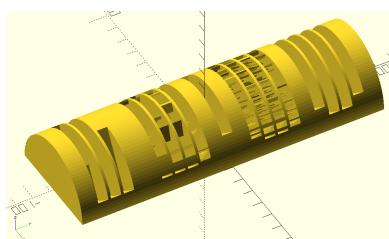


Figura 31.14: Antonia demuestra cómo importar un archivo en formato STL dentro de una escena.

»En el ejemplo de la figura 31.14, 'reloj-de-sol.stl' es el nombre que le dimos al archivo cuando lo grabamos con  —aclaró Antonia.

—Sí, me doy cuenta —Cecilia pudo verlo de inmediato, y recobró el teclado dispuesta a estampar sobre los bordes del reloj la frase de Antonia. Se valió del modificador '#' para poder ver los textos a medida que los movía; después de unos minutos de tanteos, logró un resultado que la satisfizo.

```
1 difference(){  
2     import("reloj-de-sol.stl", convexity=3);  
3     #translate([0,largo_reloj/2-borde+2.5,0])  
4         rotate([0,-90,-90])  
5             texto_circular(texto_1,5,2,  
6                 radio_semicilindro-1.2,165);  
7     #translate([0,-largo_reloj/2+2.5,0])  
8         rotate([0,-90,-90])  
9             texto_circular(texto_2,5,2,  
10                 radio_semicilindro-1.9,165);  
11 }
```

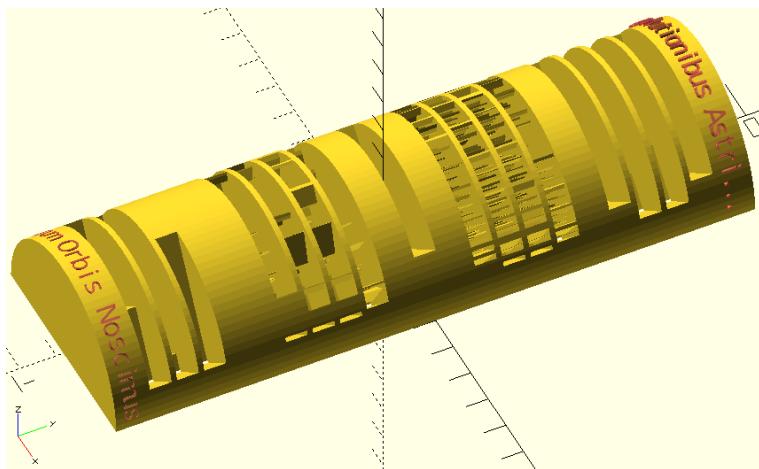


Figura 31.15: Cecilia estampa en los bordes del reloj las leyendas de Antonia.

En la línea 2, por consejo de Antonia, incluyó el parámetro

`convexity=3`: resultaba misteriosamente necesario para que el reloj no se viera como en la figura 31.16.

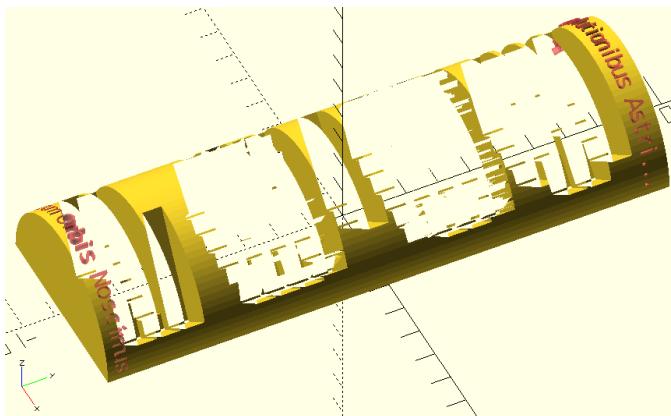


Figura 31.16: Si se emplea la sentencia `import` sin dar a la opción `convexity` un valor mayor a 1, los objetos importados pueden manifestarse de forma anómala.

Las líneas 4 y 7 servían para rotar las leyendas a fin de que discurrieran en torno al eje longitudinal del reloj; las líneas 3 y 6 las trasladaban a los bordes del mismo. El radio de cada texto era igual a `radio_semicilindro-1.9` porque el espesor de cada letra era igual a 2, y Cecilia quería que sobresalieran apenas del cuerpo del reloj a fin de que la sentencia `difference` no se encontrara frente la dificultosa ambigüedad de las superficies coincidentes.

—Perfecto —aprobó Antonia—. Ahora que comprobamos que funciona, debemos incluir ese código en el módulo `reloj_de_sol`.

Cecilia estaba completamente de acuerdo.

```

1 module reloj_de_sol(vector_horas){
2     difference(){
3         // Reloj
4         if(vector_horas==undef)
5             reloj_de_sol_continuo();

```

31. LAS ÚLTIMAS PINCELADAS

```
6     else
7         reloj_de_sol_discreto(vector_horas);
8     // Textos
9     translate([0,largo_reloj/2-borde+2.5,0])
10    rotate([0,-90,-90])
11    texto_circular(texto_1,5,2,
12        radio_semicilindro -1.2,165);
13    translate([0,-largo_reloj/2+2.5,0])
14    rotate([0,-90,-90])
15    texto_circular(texto_2,5,2,
16        radio_semicilindro -1.9,165);
17 }
```

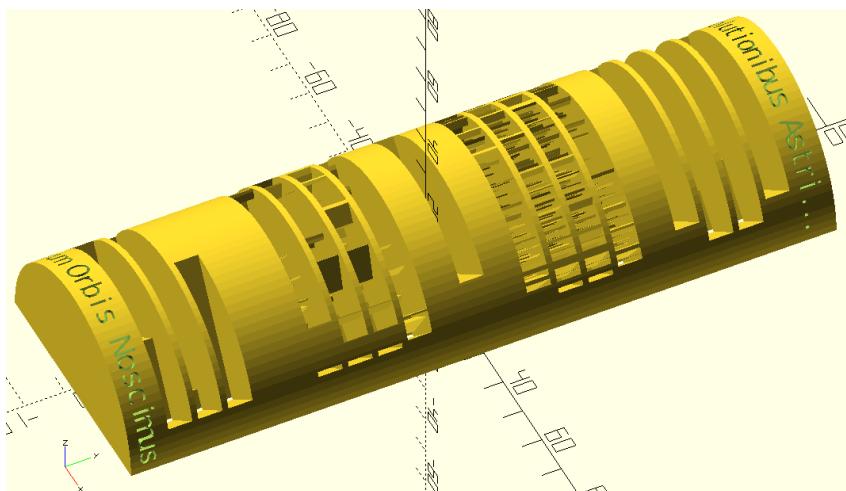


Figura 31.17: Reloj de Sol con las leyendas impresas en los bordes.

31.7. FIRMA

Las últimas medialunas habían desaparecido ya a través del tracto digestivo de ambas amigas, dejando tras sí como únicos testigos unas cuantas miguitas indiscretas en el escritorio.

Cecilia miraba su reloj de Sol con una serena felicidad, que no supo si atribuir a la dicha de haberlo concluido exitosamente o a la espléndida digestión.

Antonia, por su parte, no parecía ahora compartir su felicidad; antes bien, una sombra de melancolía ensombrecía su rostro. Casi con desgano, propuso lentamente:

—Nuestra vanidad no sería colmada si renunciamos a firmar un objeto tan soberbio como nuestro reloj —y agregó unas últimas líneas al texto que comenzaran a escribir tantos capítulos atrás, sin que ya las teclas sonaran bajo sus dedos con la alegría pasada.

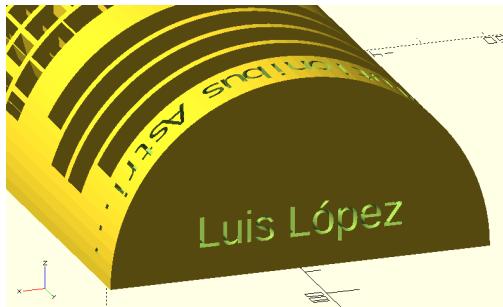


Figura 31.18: La firma del autor.

Cecilia arqueó las cejas con extrañeza ante la figura 31.18.

—¿Y eso? —preguntó.

Antonia tardó un instante en responder; su voz tenía un tono opaco:

—La firma del autor.

Cecilia se irguió lentamente en su silla. Un incierto malestar comenzó a recorrer su cuerpo; casi temió preguntar:

—¿A qué te referís, Antonia? No entiendo la broma —y trató de sonreír, pero notó que le costaba.

Su amiga giró hacia ella con lentitud antes de mirarla fijamente a los ojos:

—Este reloj lo escribió Luis, tu profesor de Astronomía —y ante la muda mirada de Cecilia, prosiguió—: Este es un manual mediocre que se propuso escribir para dejar constancia de su obra, nada más. Nosotras no existimos.

Cecilia sintió que un sudor frío se apoderaba de su frente. Quiso reír, pero nuevamente algo la contuvo. Fue como si todo su íntimo ser conociera desde siempre el significado de lo que, a todas luces, no debería haber sido otra cosa que una broma estúpida de Antonia

—No es gracioso —fue todo lo que pudo balbucear, nerviosa.

—No estoy diciendo que lo sea: la realidad no tiene la obligación de ser graciosa. Pero es real —Antonia pareció agitarse; con un tono ligeramente vibrante, añadió—: Y vos lo sabés, Cecilia: ¿Porqué durante todo este tiempo hicimos referencia, tanto vos como yo, a ‘capítulos’? ¿Cómo hice para ‘leerte la mente’ tantas veces? —se desplomó contra el respaldo de su silla, y con voz nuevamente calma, agregó—: Somos personajes, Cecilia. Simples personajes. Ni siquiera bien construidos; somos casi de cartón. Nosotras... no existimos^{2,3} —y estas últimas palabras escaparon de su boca como un suspiro.

La mente de Cecilia se vio arremetida por un huracán de incertidumbre y rebeldía; quería indignarse con su amiga, agitarse contra ella. No quería aceptarlo: no podía ser verdad. Con gran esfuerzo trató de serenarse. «Cecilia, sos una científica» —se dijo a sí misma—. «Apelá a tu capacidad de demostración».

Respirando hondo, argumentó:

²¿Yo tampoco..? (Nota del Editor)

³Vos tampoco: ¿quién hallará editor para este manualcito?

—Mirá, Antonia —procuró que su voz sonara calma; no lo logró—. Acá estoy: mirame. Existo. E-xis-to.

Apenas pronunciadas estas palabras se sintió tonta al descubrirse tratando de demostrar su existencia separando en sílabas un verbo. Con inquietud recorrió la oficina con la mirada: sus ojos se detuvieron en el vacío dejado por las medialunas.

—¡Acabamos de comer medialunas! —proclamó con tono triunfal—. ¿Los fantasmas comen medialunas, acaso?

Antonia le devolvió la mirada con una sonrisa triste y dulce:

—¿De qué eran las medialunas? —y su voz era apenas un susurro.

Cecilia sintió que una mano helada se cerraba sobre su corazón:

—¿Cómo de qué eran? —preguntó con voz quebrada, aunque entendía perfectamente el significado de la pregunta.

—Sí, de qué eran: ¿de grasa o de manteca? —en la voz de Antonia flotaba una delicada ternura.

Cecilia cubrió su rostro con las manos:

—Basta.... basta... —suplicó.

Un silencio pesado envolvió a las amigas. El Sol comenzaba a ponerse, y sus débiles rayos ingresaban por la alta...

—¡Basta, dije! —gritó Cecilia—. ¡Esto no puede terminar así! ¡Manual o no manual, esto no puede terminar así!

El grito de Cecilia me molestó menos que su redundancia: ¿No podía haber utilizado el verbo “concluir” en la tercera exclamación? Como sea, decía que el Sol comenzaba...

—¡NO! —Cecilia se puso de pie en medio de la oficina—. ¡Dije que no, Luis!

Esto no está concluyendo como esperaba, evidentemente. Antonia también se levantó; espero que tranquilice a Cecilia y podamos cerrar el capítulo y el manual como corresponde y en paz.

—No es tan malo como parece —Antonia apeló a la empatía para sosegar a su amiga—. Ya nos ocurrió con el manual del ApNE,

y aquí estamos. Y seguramente volveremos a estar en algún otro manual tan malo como estos.

Cecilia se agitó con cierta violencia; esta vez parecía pasar algo más:

—¡No! ¡Me niego! —exclamó con firmeza; en su voz ya no había nerviosismo—. Quizá no pueda demostrarlo. Quizá sea más natural aceptar que somos un manojo de palabras. Pero yo sé que no es así: yo existo —pareció morder esta última palabra; tras girar bruscamente hacia su amiga, añadió—: Y vos también, Antonia: vos existís.

Confieso que a esta altura ya estoy inquietandome un poco; más que nada porque estamos en la página 26 del capítulo y quería ir cerrándolo. Pero debo admitir también que se ha despertado en mí una cierta sensación de divertida hilaridad. Esta situación incluso me recuerda algo...^{4,5}

Cecilia caminaba por la oficina; sus pasos sonaban con firmeza contra el piso de madera. Sus ojos brillaban con el fuego de un pensamiento intenso y voraz. Antonia, inmóvil, la seguía con una mirada dulce y comprensiva.

Cecilia detuvo de golpe sus pasos:

—¡Ya sé! —exclamó—. Luis sigue trabajando aquí, en Harvard, ¿no? —y en su mirada una luz de feroz intensidad se abrió paso.

—Sí; en el Observatorio —contestó Antonia—. ¿Por qué? — preguntó a continuación y con cautela.

Cecilia no respondió inmediatamente. Respiró con fuerza, y dirigiéndose con decisión hacia la puerta aseguró:

—Para hacerle una visita. ¿Venís?

Y antes de que pudiera terminar el capítulo, juntas abandonaron la oficina y se dirigieron al Observatorio de Harvard, donde acabo de enterarme de que las estoy esperando.

⁴La novela *Niebla*, de Miguel de Unamuno! ¡Plagio! (nota del Editor)

⁵Novela, no: nivola. Pero es verdad: me recuerda ese libro. Y no seas resentido: no es plagio. Digamos que es un... ¿homenaje?

Epílogo

DEBO ADMITIR QUE me invade una sensación de incómoda perplejidad. Si bien no puedo ocultar que la idea de conversar con Antonia y Cecilia me atrae desde hace tiempo —para no mencionar, claro, la dulce caricia a mi vanidad que representa aparecer en primera persona dentro de este manualcito—, lo cierto es que no tengo la menor idea de lo que voy a decirles. Y, por supuesto, tengo aun menos claro que es lo que ellas quieren decirme. Me preocupa sobre todo Cecilia, a quien parece que no dejé precisamente de buen humor en el último capítulo. Supongo que lo mejor será dejar en manos de ambas el hilo de este epílogo: casi no hice otra cosa, por lo demás, en los capítulos precedentes.

Pero ya mi perplejidad parece a punto de terminar: el sonido familiar de pasos en la escalera de entrada al Observatorio interrumpe mi soliloquio inicial. Como siempre que esos pasos anuncian la llegada de una visita me levanto del sillón de madera del escritorio, aliso maquinalmente los pliegues de mi camisa con las manos y me bajo de la tarima que preside el aula con la mirada puesta en la puerta y una sonrisa amistosa flotando en los labios.

No me sorprendió ver aparecer primero a Cecilia, seguida inmediatamente por Antonia. Las pocas fotografías que había visto de ellas me permitieron reconocerlas enseguida: los fotógrafos

de principios del siglo XX eran perfectos retratistas, a los que los fraudulentos programas de procesamiento de imágenes actuales no habían privado, como lo consiguen ahora plenamente, de la capacidad de capturar con fidelidad el carácter y la belleza de estas dos mujeres.

Las dos se detuvieron en cuanto traspasaron el umbral de la puerta del aula, fijando en mí sus miradas. Los tres nos observamos en silencio unos instantes. Preveo que será Cecilia quien inicie la conversación.

—Hola, Luis. ¿Cómo estás? —dijo, confirmando mi sospecha. Su voz era firme. Tal vez demasiado: intuí en esa firmeza un tanto enfática que estaba tan nerviosa como yo.

—Muy bien, ¿y ustedes? —procuro parecer tranquilo; no me cuesta mucho, tampoco: de pronto recuerdo que soy el autor de este manual y nada puede ocurrir en él fuera de mi voluntad. Considero apropiado evocar algunos recuerdos comunes—. ¡Tanto tiempo, Cecilia! Vos fuiste mi alumna: imposible olvidarlo. Solías sentarte en... ese banco —señalo con seguridad, confiando en no equivocarme—. A vos, Antonia, no tuve el gusto de conocerte personalmente. Pero, ¿quién no te conoce aquí, en Harvard? Es un verdadero placer.

El gesto de Antonia se ensombreció ligeramente: ¿habrá pensado que la conozco por supuestas malas habladurías que corren sobre ella? Olvidé que es muy susceptible y vulnerable. Procuraré tenerlo más en cuenta.

Cecilia, por su parte, recibió mis palabras con un gesto radiante de satisfacción; dirigiéndose a su amiga preguntó, como quien lanza un desafío:

—¿Ves, Antonia? ¡Luis nos conoce! Yo fui su alumna, y vos sos una celebridad en esta institución. Ergo, existimos. El asunto queda zanjado y creo que ya podemos retirarnos —concluyó con tono marcadamente triunfal, y a punto de dar media vuelta. Siento un gran alivio: el manual terminará sin mayores sobresaltos y en

paz.

Pero ahora fue Antonia quien parecía querer estirar esta incómoda situación un poco más:

—Cecilia, siempre fuiste muy ingenua. Es un milagro que lograras ser una gran científica confiando así en los demás —se detuvo un instante, y continuó—: Aunque, pensándolo bien, una científica debe dudar de la naturaleza y sus leyes, más que de las personas. Como sea, ¿no te das cuenta de que Luis sólo simula haber sido tu profesor, como lo hizo en el capítulo 23?

Cecilia la miró entre sorprendida y ofuscada: había olvidado que Antonia era difícil de convencer. Me temo que deberé continuar unas cuantas páginas más.

Antes de que su amiga pudiera replicar, Antonia se apresuró a reforzar su posición:

—A ver, ¿recordás alguna pregunta que le hayas hecho en clase?

En este punto siento que puedo intervenir a favor de Cecilia:

—Bueno, Antonia; si es por eso, podría decirse que no tuve alumnos de ningún tipo: ¡casi nadie me hace preguntas en clase!

Cuando se me ocurrió esta idea me pareció graciosa: apenas escrita, advierto que es más bien melancólica.

Cecilia giró lentamente su mirada en mi dirección; en sus ojos flotaba una luz que parecía de venganza:

—A ver el gran profesor que sí existe: ¿Ahora resultará que él también pertenece a un mundo de ensueño e irrealdad?

Las palabras de Cecilia me toman por sorpresa. ¿De dónde habían salido? ¿Se convertirá este epílogo en una confesión de mal gusto, con tintes psicológicos de segunda o tercera categoría? Debo evitarlo como sea.

—Por supuesto que no —respondo como sin darle importancia al asunto, y sonrío con la mayor suficiencia que puedo: tal vez hasta parecer grosero—. Como todo adolescente pasé por mi período de solipsismo, desde ya —me parece necesario recono-

cer—; pero lo superé satisfactoriamente.

Cecilia parecía interesada en mi reciente confesión:

—¿Y cómo hiciste para superarlo?

No era muy difícil entender su interés: seguramente querría aferrarse a mi “método” para emplearlo ella misma en la demostración de su propia existencia. Ante ese evidente y fútil intento, esta vez no tuve que forzar mi sonrisa.

—Pues... supongo que como todos —digo, tratando de evocar recuerdos bastante lejanos, tanto en el tiempo como en el conjunto de mis preocupaciones actuales—. Un poco por cansancio y un poco por obligación: sentí que debía actuar en función de los datos a los que me enfrentaba. Aun si sólo fuera una mente solitaria pensándose y pensando un Universo fantasmal, esa misma mente me lo representaba de una manera específica, a la cual sentía mi deber adherir y responder, aunque fuera de manera reactiva; pero nunca negándolo —concluyo, con el mayor aplomo que puedo.

—¡Ajá! —lanzó Cecilia desafiante—. Puedo invocar el mismo principio, entonces: yo percibo el mundo y mi existencia en él; luego, debo aceptar que existo y actuar en consecuencia.

—Hay una gran diferencia, Cecilia —señalo suavemente—. Yo puedo ejercer, o en todo caso soñar que ejerzo, mi libre albedrío: vos no.

Cecilia empalideció ligeramente.

—¿Qué querés decir? —preguntó bajando el tono de voz.

—Pues... yo puedo hacer que digas, ahora, la palabra ‘elefante’ —prometo, sin pretender ser demasiado original.

—¡Imposible! —se rebeló, pero enseguida añade: —Elefante.

Giró rápidamente el rostro en dirección a Antonia, desconcertada; pero sólo encontró a su amiga devolviéndole la mirada con gesto de tranquila resignación.

—Elefante. Elefante. Elefante —repite tres veces. Cecilia se desplomó en uno de los bancos del aula; cubriéndose el rostro con las manos, susurró:

—Está bien; entendí. Ya basta, por favor.

De pronto me siento muy cruel; tal vez no era necesario ser tan categórico. Pero, ¿qué podía hacer? Este manual debe terminar alguna vez, después de todo. No es que esté justificándome, tampoco; no tengo porqué hacerlo, lector.

Como sea, ahora no se me ocurre mejor cosa que invitar a Antonia, con un gesto conciliador, a sentarse en otro banco mientras yo rodeo el escritorio para ocupar su correspondiente sillón.

Busco las palabras para tratar de suavizar la situación:

—Cecilia, seamos sensatos. Ustedes *tienen* una existencia real.

Los ojos de ambas se abrieron con sorpresa; con una mano en alto interrumpí posibles objeciones:

—No me malinterpreten; no me refiero a su existencia en el marco de este manualcito. Como bien dijo Antonia en el capítulo anterior, ustedes aquí son mis personajes: bastante mal construidos, por cierto. ‘Casi de cartón’ fue la expresión que usaste, si no me equivoco.

Tomé el silencioso asentimiento de Antonia con un movimiento de cabeza como una invitación a continuar:

—Pero tienen una existencia propia innegable: las dos serán recordadas por siempre mientras la Astronomía siga siendo una actividad vital de investigación y cuestionamiento constantes, tanto del mundo que nos rodea como de nuestras propias inquietudes y miserias: inquietudes por conocer y entender, pero también miserias al relegar y negar las capacidades de algunos grupos humanos, ya sea por motivos de género o de cualquier otra índole —me dejo llevar por un cierto tono retórico que no quiero interrumpir; procuro que mi voz sostenga una contenida elevación apropiada al final del período que ya presento—. Ustedes existieron y existirán por siempre: para mostrarnos lo que debemos hacer, lo que somos capaces de hacer, y lo que no debemos permitir que nadie haga nunca más.

Hago un silencio teatral y las miro esperando que transformen

sus rostros en soles radiantes de felicidad y gratitud. Sin embargo, noto que la expresión de ambas adquiere un tono reconcentrado, y de cierto interés y preocupación. De pronto siento que no sólo me miran, sino que me observan y analizan.

Esta vez fue Antonia quien retomó el hilo de la conversación:

—Lo que nos contaste ya lo sabemos, Luis: Nosotras no existimos en este manual, pero fuimos grandes astrónomas y seremos recordadas por siempre.

Tras intercambiar una mirada con Cecilia, quien pareció asentir tácitamente a un pensamiento común, bajó la vista como si buscara la mejor manera de continuar; inmediatamente agregó, clavando en mí su mirada:

—¿Por qué nos trajiste aquí? —y en su voz vibraba una nota de delicada ternura, como quien habla con un niño a quién se desea hacer sentir comprendido.

Me siento raro; es como si me hubieran descubierto en un delito cuya naturaleza ignoro. Sólo alcanzo a repreguntar:

—¿Cómo por qué? Supongo que me pareció una buena idea. No sé... Algo gracioso. Ya conocen una de mis citas favoritas: 'Toda labor intelectual es, en última instancia, humorística'.

Sé muy bien que no es ésa la respuesta que esperan; alzando las cejas las invito a continuar, no sin cierta aprensión.

Cecilia suspiró con fuerza, e intervino con una expresión de paciencia contenida:

—Vos sabés que ésa no es la única razón. Repetiré la conclusión anterior, para ser más clara: Antonia y yo no tenemos existencia aquí, pero vivimos hace tiempo en Harvard y seremos recordadas por siempre —e hizo silencio, mientras ambas me miraban como quien sólo espera una conclusión evidente, la cual debo confesar que se me escapa. Sólo puedo alzar los hombros, admitiendo mi perfecta incomprendición.

Antonia se mordió ligeramente los labios, y casi a su pesar agregó con suavidad:

—Por otra parte, vos sí existís en el marco de este manual. Sólo ahora entiendo lo que querían decirme; es un baldazo de agua fría, tan repentino y violento como un relámpago. Me había propuesto no convertir este epílogo en una confesión psicológica de segunda categoría, y aquí me encuentro enfrentado por mis propios personajes a la melancólica e innegable realidad: yo no seré recordado, ni dejaré huella alguna permanente en la memoria de la humanidad. En cierto sentido profundo y verdadero, son ellas quienes tienen una existencia mucho más real que yo.

Me desplomo lentamente contra el respaldo del sillón de madera del escritorio. Mi mirada se pierde, ausente, a través de las ventanas del fondo del aula. «¿Para esto las hice venir?» —me pregunto—. «¿Era necesario todo un epílogo que ya cuenta con 7 páginas sólo como torpe exégesis de los suficientes versos de Borges que no puedo dejar de copiar cada vez que tengo la oportunidad de hacerlo..?»

Un poeta menor

La meta es el olvido.
Yo he llegado antes.

Quince monedas, Jorge Luis Borges (1975).

El sabor de esos versos, tantas veces repetidos, vuelve a recordarme la miel y la resignación. En medio de esa dulce melancolía casi me olvido de mis compañeras, hasta que mis ojos tropiezan inadvertidamente con los de Cecilia: me parece ver en ellos una luz apagada de pena. ¿Será ella, ahora, quien se sienta mortificada por haber sido tal vez demasiado cruel? ¿Deberé tolerar unas palabras de consuelo por parte de mis propios personajes? Eso sería demasiado patético.

Fue Cecilia quien comenzó; no puedo decir que me sorprenda.

—La labor de un profesor es muy importante —declaró con tono suave y seguro—. Tiene en sus manos la capacidad de despertar en las personas la curiosidad, el anhelo por conocer, por superarse.

Como docente me resulta imposible dejar de ver el reverso de la trama del incipiente discurso de Cecilia; supongo que fue mi gesto de desagrado lo que la detuvo en seco. Dirigió a su amiga una mirada que era un pedido indisimulado de auxilio.

Antonia se acomodó mejor en su pupitre, mientras parecía buscar las palabras adecuadas:

—Lo que queremos decir, Luis —dijo con cautela y cierta lentitud—, es que hay muchas maneras de perdurar en la memoria de los demás.

—Claro: se puede lograr siendo una gran científica cuyas obras se impriman, divulguen, expliquen y analicen, o siendo un profesor en una escuela secundaria —interrumpo mordazmente, sin sentirme siquiera de humor para conservar al menos una apariencia superficial de buena educación.

Cecilia y Antonia apretaron los labios; parecen sentir que me estoy poniendo difícil. Y bueno; ellas quisieron venir, después de todo. Podrían haberse quedado tranquilamente en el capítulo 31, como era mi intención original. Ahora que se embromen.

Cecilia se agitó ligeramente, recobrando su aplomo de siempre. Eso me gusta: supongo que no me espetará más lugares comunes.

—Luis, no seas injusto. No sólo con vos: tampoco con los demás. Varios de tus alumnos tienen palabras muy elogiosas respecto a vos —aseguró, y con tono que descartaba toda objeción, agregó—: No vas a sugerir que son falsos; eso sí sería muy cruel, sobre todo para con ellos. La modestia es una virtud, pero la desconfianza está muy cerca del insulto.

Las palabras de Cecilia me descolocan, debo admitirlo. Trato de advertir en ellas un intento de manipulación, pero el vigor con que las expresó de alguna manera me convencen de su sinceridad.

Me levanto del sillón y paseo por el aula, entre el escritorio y los bancos: señal inequívoca de que necesito pensar.

De pronto siento que lo encontré:

—Ok. Acepto los elogios y su sinceridad —considero más convincente empezar reconociéndolo—. Pero, ¿qué significa exactamente eso? —hago una pausa teatral para dar más fuerza a lo que sigue—. ¿Por qué me elogian? ¿Porque soy un buen profesor? ¿Porque soy una buena persona? ¿Porque les caigo bien? ¿Porque...?

Esta vez fui yo quien se detuvo bruscamente al ver el gesto de ambas amigas. Sí; tal vez fui demasiado lejos. Ahogo un suspiro mientras me apoyo en la baranda de la escalera de acceso a la cúpula.

Los tres mantenemos un silencio pesado; nuestras miradas ya no se cruzan. Mis ojos advierten sin interés alguno los rayos de Sol sobre las baldosas del aula: no me resultan cálidos, ni brillantes, ni nada.

Vuelven a oírse pasos en la escalera: esta vez son veloces, inquietos. El timbre del recreo acababa de sonar; seguramente es un alumno en una visita fugaz.

Es Lucy; una sonrisa acude sin dificultad a mi rostro. Fue alumna mía hace muy poco: compartí con ella, como con todos, inquietudes y problemas astronómicos y de índole diversa. Subió los escalones de dos en dos; estaba visiblemente apurada. No había alcanzado la puerta y ya me saludaba con entusiasmo y alegría:

—¡Luisss!

Me extraña que pase al lado de Cecilia y Antonia como si no las viera, dándoles olímpicamente la espalda: Lucy fue siempre muy educada. De todas formas, no me da tiempo siquiera para presentarlas:

—Estuve en una clase muy aburrida —soltó con tono vibrante y veloz—. Se me ocurrió una manera muy loca de generar

números con una fórmula; mirá.

Puso ante mis ojos una hoja de carpeta cargada de símbolos y tachaduras, que aprovechaban todo su espacio sin respetar los euclídeos caminos de sus horizontales renglones. Traté de entenderlos, mientras ella se transformaba en un volcán de explicaciones y aclaraciones. Tomé en mis manos esa hoja: era desplolija, secreta, redundante, inconclusa, oscura, titubeante, explosiva. Era, en una palabra, hermosa.

El timbre volvió a sonar urgentemente. Lucy recuperó su hoja de entre mis manos y con un sonoro «¡Gracias, Luis! ¡Vuelvo más tarde!» se dirigió velozmente a la puerta. Apenas tuve tiempo de preguntar inocentemente:

—¿Para qué es la fórmula? ¿Querés que te ayude en algo?

Lucy dio media vuelta, extrañada. Desde el umbral de la puerta respondió:

—No, nada; es algo que se me ocurrió y te lo quería mostrar. ¡Chauuuu..! —y bajó por la escalera con la misma rapidez con la que subió, sin tampoco despedirse ahora de Antonia y Cecilia.

Cuando el sonido de los pasos de Lucy se perdió a lo lejos, dirigí a ellas la mirada. Estaban sonriéndome con ternura; en sus ojos flotaba una luz de íntima satisfacción. Mientras me encontraba en el reflejo de esos ojos repasé mentalmente lo ocurrido: una alumna había subido al Observatorio a mostrarme un personal desarrollo matemático tal vez sin sentido y para el cual no tenía otra motivación que la pasión por realizarlo.

De pronto me sentí como vencido; debo haberme ruborizado, incluso.

Quise decir algo, pero no supe qué. No fue necesario; Cecilia y Antonia se levantaron lentamente, sin hacer el menor ruido. Se dirigieron hacia la puerta, desde donde se despidieron moviendo levemente sus manos en alto. Sus sonrisas seguían diciéndome desde el umbral que ellas, finalmente, tenían razón.

No oigo sus pasos a medida que se alejan por la escalera. Ya

no me extraña ni me deja de extrañar.

—Adiós, queridas amigas.

—Y... muchas gracias.

Nuevamente a solas, mis ojos recorren con gratitud y cariño el familiar espacio del aula vacía. Los pupitres, apenas ordenados, son otras tantas promesas que florecerán, sin duda, el año que viene.

Encuentro en los rayos del Sol que entran al aula toda la luz y la calidez que hasta hace un momento habían perdido. Las ventanas y los bancos sólo permiten que algunos de esos rayos toquen las baldosas, escribiendo en ellas con símbolos secretos que ningún algoritmo será capaz de repetir.

Suspiro profundamente. Estoy por volver al sillón de madera cuando una chispa se enciende bruscamente en mi alma; salgo disparado hacia la puerta mientras procuro que mi voz alcance a oírse:

—¡Esperen! ¡Se me acaba de ocurrir una idea...!

FIN

APÉNDICE

— A —

Sólo un poco más de Astronomía

CECILIA Y ANTONIA se alejaban del Observatorio con paso tranquilo y ligeramente melancólico, recorriendo una vez más los interminables pasillos de Harvard. El Sol las acariciaba cálidamente a medida que pasaban frente a los amplios ventanales que iluminaban los claustros. El silencio que las acompañaba era suficientemente elocuente, y las dos se dejaron arropar por él, por su amistad y por la luz solar.

De pronto Cecilia detuvo sus pasos:

—¿Escuchaste algo? —preguntó.

Antonia se detuvo a su vez y prestó atención; tras unos breves instantes respondió:

—No, nada... ¿Por?

Cecilia sacudió la cabeza levemente; con una sonrisa retomó la marcha:

—Por nada; me pareció sentir que nos llamaban.

No dieron más que unos cuantos pasos cuando ahora fue Antonia quien se detuvo. Se encontraban en ese momento al lado de uno de los ventanales. Giró sobre sí misma para enfrentarse a él. Tenía los ojos cerrados; pareció disfrutar de la caricia del Sol sobre sus párpados mientras respiraba profundamente. Cecilia la contemplaba con ternura: no recordaba haber visto antes a su

amiga disfrutar de las sensaciones delicadas y elementales que nos regala la naturaleza.

Antonia abrió los ojos y exhaló un fuerte suspiro. De las profundidades del morral que siempre la acompañaba extrajo el reloj de Sol digital que había impreso antes del comienzo de esta aventura. Cecilia lo reconoció inmediatamente: era el mismo que vio en su oficina en el capítulo 2, y con cuyos rayos la asombró en el capítulo primero.

—Me gustaría que discutiéramos algo —propuso Antonia.

A.1. UN *bug* MÁS BIEN ASTRONÓMICO

—En el capítulo 23 concluimos que el cuerpo del reloj debe colocarse de forma paralela al eje de la Tierra —evocó Antonia—; en otras palabras, inclinado un ángulo igual a nuestra latitud y dispuesto en el plano vertical que une los puntos cardinales Norte y Sur.

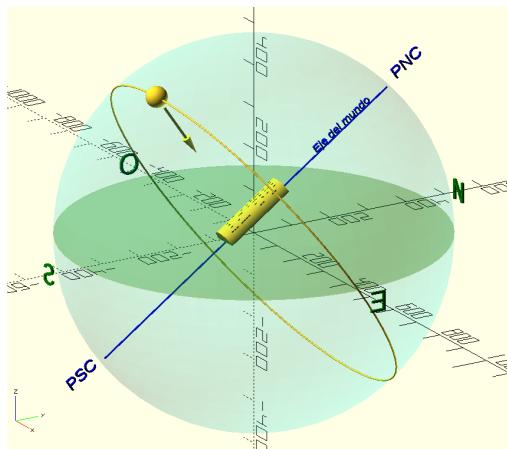


Figura A.1: Antonia y Cecilia concluyeron en el capítulo 23 que el reloj debe orientarse paralelamente al eje terrestre.

Cecilia lo recordaba perfectamente.

—La latitud de Harvard es de unos 35° Sur¹ —continuó su amiga—; puse una marquita en el soporte para poder inclinar el cuerpo del reloj con certeza.

Con movimientos pausados, casi ceremoniosos, Antonia se inclinó hacia el piso para ubicar el reloj de Sol digital orientado de la manera en que habían deducido que debía colocarse. Sin embargo, para sorpresa de Cecilia ningún dígito se formó debajo del mismo; miró a su amiga sin poder disimular un gesto de alarma:

—¿Qué pasó? —preguntó, visiblemente consternada ante la figura A.2.



Figura A.2: El reloj de Sol, para alarma y angustia de Cecilia, no indica la hora.

—Tranquila —contestó Antonia con una sonrisa—. Se trata de un pequeño detalle que yo tampoco tuve en cuenta la primera vez que probé el reloj frente al Sol, y sobre el que no quise llamar tu atención en el capítulo 23... ¡Demasiados problemas teníamos

¹Ya entendí todo. (Nota del Editor)

por delante, y de índole más urgente y teórica, antes que éste, de naturaleza apenas ‘práctica’ y de muy fácil solución..!

La alarma de Cecilia fue reemplazada por una suerte de indignada sorpresa: ¿El hecho de que el reloj no mostrase la hora le parecía a Antonia un detalle menor y de índole meramente “práctica”? Cecilia conocía ya de sobra la inclinación de su amiga por las ideas “puras” y la teoría; inclinación ésta que a veces la llevaba a mirar con desdén todo aquello que supusiera una aplicación o resultado concreto y material. Pero esto ya le parecía el colmo.

Antonia, indiferente a la sorpresa de su amiga, continuó disertando:

—El problema se manifiesta cuando el Sol se encuentra relativamente alejado del Ecuador Celeste —explicó—. En ese caso, los rayos solares no pueden pasar por los huecos que se encuentran óptimamente dispuestos sólo cuando el Sol se encuentra en dicho plano: en otras palabras, más bien cerca de la primavera o el otoño.

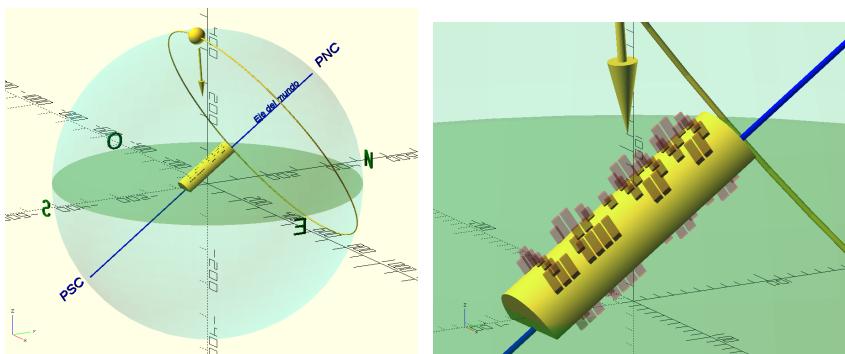


Figura A.3: Cuando el Sol se encuentra muy alejado del Ecuador Celeste (o sea, cerca del invierno o del verano), sus rayos no alcanzan a atravesar los agujeros del reloj, que fueron dispuestos para recibirlos de manera perpendicular a su cuerpo.

Cecilia pudo apreciar con claridad el problema en la figura A.3. Imaginó una forma de resolverlo y, aunque en el fondo no le

gustaba, se sintió obligada a proponerla:

—¿Aumentamos el ancho de los pixeles? De esta manera los rayos solares podrán atravesar los agujeros, al menos de soslayo, aun en los solsticios.

Antonia frunció el gesto; a ella tampoco parecía gustarle esa solución.

—Lo que proponés no es, por supuesto, ineficaz —comentó—. El inconveniente es que eso nos obligaría a alargar mucho los pixeles... Te confieso que no hice las cuentas, pero estoy segura de que resultaría un reloj de Sol demasiado largo. Tal vez ni siquiera podríamos imprimirlo.

Cecilia pensó que siempre podían imprimir el reloj por partes y luego pegarlas; pero prefirió dejar que Antonia se tomara su tiempo para ofrecer la solución que, sin duda, consideraría óptima.

—Además, si alargamos los pixeles obtendremos sobre el pi-
so horas muy anchas en los equinoccios y más angostas en los solsticios —añadió Antonia, como si necesitara refutar del todo la propuesta de Cecilia antes de adelantar la suya propia. Cecilia pensó que ésta quizá no sería tan buena, después de todo, si antes era preciso destacar tantos defectos en la otra.

Antonia hizo un breve silencio mientras miraba el reloj de Sol digital con gesto reconcentrado; parecía buscar la mejor manera de presentar su solución.

—Mirá, para mí, lo mejor es simplemente modificar, los días que resulte necesario, el ángulo de inclinación del reloj para que los agujeros resulten paralelos a la dirección de los rayos solares —propuso finalmente y con el aire de seguridad y suficiencia con el que soltaba las ideas que, en el fondo, sentía que no podía justificar del todo: otro más de sus defectos docentes.

Cecilia se cruzó bruscamente de brazos: se sentía francamente decepcionada.

—¿Modificar la inclinación del reloj? —repitió en forma de pregunta, sin disimular un tono de frustración.

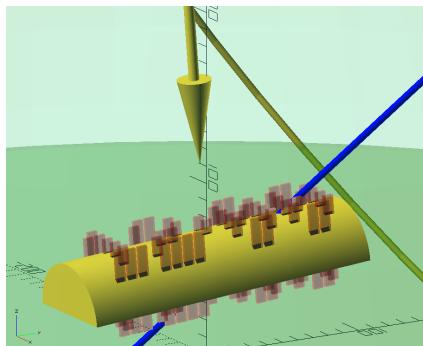


Figura A.4: Antonia propone modificar la inclinación del reloj para que su cuerpo resulte, cada día, perpendicular a los rayos solares.

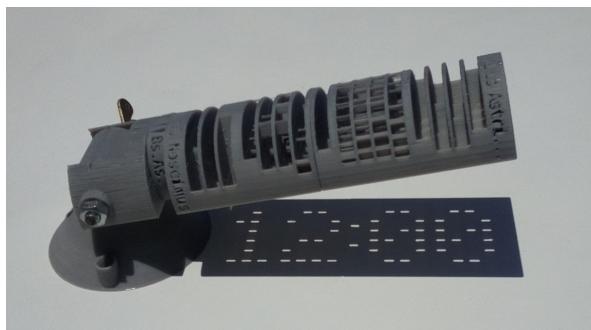


Figura A.5: El reloj de Sol, debidamente inclinado, indica la hora aun en los solsticios.

—Sí —replicó Antonia, alzando cejas y hombros como si hubiera propuesto la solución más natural del mundo.

Cecilia mantenía los brazos cruzados, y ahora tamborileaba con la punta del pie contra el piso. Podía ver que la solución funcionaba, pero le resultaba un tanto chapucera. Tanto ella como Antonia mantenían los ojos fijos en el reloj, que silencioso e

indiferente mostraba la hora con puntualidad solar.^{2,3}

² ¿Y? ¿Ya está? ¿Pero ahora qué hacemos? (Nota del Editor)

³ Nosotros, nada: si después de 377 páginas no fuimos capaces de permitir a nuestro querido e improbable lector que encuentre sus propias respuestas por sí mismo, triste librito sería éste. (Nota de Antonia, Cecilia y Luis)

Notas