Recibido: domingo, 28 octubre 2007

multimedia ::
matematicalia
revista digital de divulgación matemática
Vol. 3, nos. 4-5 (oct.-dic. 2007)

Demoscene: arte, creatividad y matemáticas en movimiento

Iñigo Quílez
VRcontext (Bruselas)
e-mail: iquilezles@hotmail.com
página web: http://www.rgba.org/iq

Introducción

En este artículo voy a intentar dar a conocer algo que creo es extraordinario y fascinante, y que al mismo tiempo es desconocido a la práctica totalidad de la sociedad. De hecho, resulta inquietante que, tratándose de un tema que no requiere de grandes conocimientos ni estudios sino que, al contrario, se trata de algo elemental y sencillo de comprender, haya permanecido oculto, tal vez guardado con celo, por más de veinte años. La tarea que aquí me propongo es acercarles al mundo de la "demoscene".

"Demoscene" es una palabra inglesa, que engloba todo un movimiento cultural que gira alrededor de la creación de "demos". Habría ahora que definir lo que es una "demo", pero, lamentablemente, no existe tal definición. Confío en que el significado de "demo" haya quedado claro al final del artículo, ya que la idea principal de éste no es sino dar una idea de lo que es una demo, y por tanto, la demoscene (ahora ya sin comillas). Veamos.

Cierren los ojos (mentalmente, claro). Imagínense que fuera posible crear un cuadro sólo con fórmulas matemáticas. Esperen, no los abran aún. No hablo (necesariamente) de un cuadro abstracto, o un cuadro fractal, ni nada derivado del capricho de las fórmulas más que de la voluntad del artista. Hablo de un auténtico cuadro en el sentido tradicional de la palabra, en el que las fórmulas juegan el papel de pinceles, no de pintor. Tal inusual cuadro no sería creado con acuarela ni programa informático de dibujo alguno, sino con matemáticas. Sin abrir todavía los ojos, imaginen además que este cuadro tiene la propiedad de mutar, de moverse, como una especie de obra de teatro, en la que el movimiento de los actores, cambios de escena y demás elementos dinámicos son resultado de las matemáticas implicadas en la creación del cuadro, y por tanto algo inherente al mismo. Imaginen, por último, que fuéramos capaces de acompañar tan extraordinario cuadro matemático con música creada también sólo mediante la aplicación de fórmulas matemáticas. No imaginen una música aleatoria, ni fractal, ni minimalista, sino verdadera música con melodía, y perfectamente acompasada con el movimiento del cuadro. Cuadro que muestra paisajes, o esculturas por ejemplo. Por último, imagínense que todas estas fórmulas pudieran ser tan sencillas como las que estudiantes de bachillerato aprenden en nuestras escuelas. Ahora, abran los ojos por fin y dejen de imaginar. Permítanme mostrarles este tipo de cuadros, las "demos".

Las demos

Efectivamente, las demos (ahora ya sin comillas) existen; nada más y nada menos que cerca de veinte mil en la actualidad. Se materializan en forma de pequeños programas informáticos (escritos por lo general en C) que al ser ejecutados en el ordenador muestran una obra puramente matemática. Además, las demos funcionan en "tiempo real", lo que significa que las fórmulas se evalúan sobre la marcha durante la visualización de las imágenes. En otras palabras, las demos son una lista de fórmulas que la CPU del ordenador realmente "interpreta". No es una obra de arte pregrabada en formato digital, como lo sería una imagen fractal, o un video o película, o un CD de música, sino que muy al contrario, una demo es algo vivo que palpita mientras la vemos.

Soy consciente de que es difícil dar una idea de lo que es una demo, y por tanto dar sentido al resto del artículo. Pero el problema tiene solución sencilla; propongo al lector que en este punto se detenga y descargue de Internet la siguiente demo (no se trata de una de las demos más bellas ni mucho menos, pero es ideal para explicar ciertas cosas). Una vez bajado, debe descomprimir el archivo zip con *Winzip*, y a continuación ejecutar el archivo con nombre *rgba_paradise_final.exe*. Siga leyendo cuando la demo haya terminado...

Me gustaría recalcar que lo que acabamos de ver eran fórmulas y nada más que fórmulas; no era un videojuego, ni una película. De hecho, lo que *Paradise* muestra no son ballenas sino elipsoides deformados, no son delfines moviendo la cola, sino cosenos con fases adecuadas, ni mariposas sino funciones trigonométricas polares. Lo que se asemejaba a un bosque de árboles, eran cilindros desplazados con oscilaciones y atenuación exponencial. Tampoco ha visto agua, sino ecuaciones de reflexión y refracción de luz, ni un caballo, sino algún tipo de deformación, ni un ciervo. La música

no era un MP3, sino un montón de funciones matemáticas seleccionadas adecuadamente en cada momento excitando el oído a través del altavoz del ordenador.

Si todo esto le resulta inverosímil, y los términos informáticos le son familiares, fíjese que el programa que se acaba de descargar ocupa apenas 60 kilobytes, que equivale a aproximadamente medio segundo de película de vídeo en DVD, o unos tres segundos de música en formato MP3, o menos de lo que ocuparía una fotografía digital, o incluso un documento de Word. Es decir, todo el contenido de *Paradise* ha brotado de una cantidad irrisoria de datos. Aun así, el programa ha mostrado 8 minutos de música e imágenes en movimiento. Evidentemente la receta mágica es la utilización de fórmulas matemáticas para crear sobre la marcha todo este material audiovisual.

Las matemáticas

Antes de empezar a explicar cómo esto es posible, permítanme concluir la introducción haciendo notar que existen muchos tipos de demos, tanto en estilos artísticos como en tecnología utilizada, o complejidad matemática implicada. Efectivamente, como en todo movimiento, existen estilos y tendencias que a veces marcan épocas. Además, desde que las primeras demos se hicieron en los años 80 la tecnología y, sobre todo, las técnicas han evolucionado. Por otra parte, muchas demos son híbridos entre matemáticas y disciplinas artísticas tradicionales como dibujo o composición musical. En cualquier caso, todas ellas tienen algo en común, y es el uso creativo de matemáticas, programación y diseño para la creación de una pequeña obra.

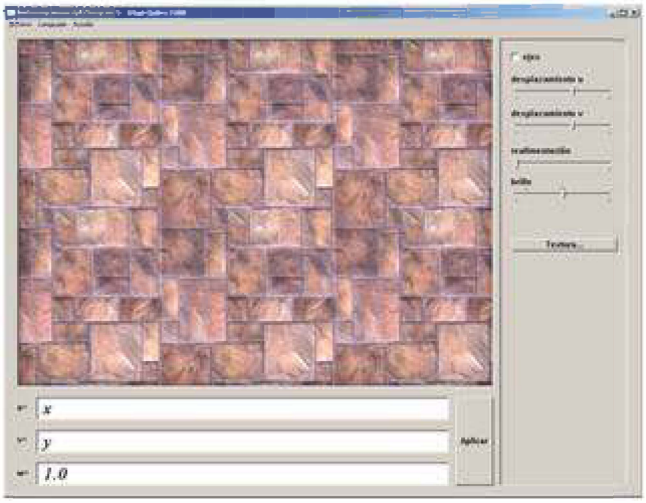
Desde el punto de vista matemático, que es lo que aquí más nos interesa, si bien es cierto que algunas demos utilizan descomposiciones de funciones en esféricos armónicos (series de Fourier esféricas) o en Wavelets, o utilizan transformadas Z para sintetizar música, una gran cantidad de demos están realizadas por adolescentes sin semejantes conocimientos. Lo que demuestra que no es imprescindible tener un alto nivel de conocimientos matemáticos para disfrutar de las matemáticas y desarrollar un trabajo creativo con ellas, como lo es una demo. Basta con tener los conceptos básicos de bachillerato asimilados, y acompañarlos de una potente interpretación, no necesariamente teórica, sino más bien intuitiva. De hecho, crear una demo no puede sino ayudar a desarrollar más estas intuiciones, y podría decirse que implica un aprendizaje involuntario de los conceptos matemáticos elementales. Enseguida lo veremos.

Jugando sobre el plano

Ahora les invito a descargarse y disfrutar una segunda demo. En este caso la música no es matemática, ni alguno de los dibujos que aparecen al final de la demo. Sin embargo, cada color y movimiento de esta hipnotizante pieza es resultado de muchas fórmulas matemáticas, que podríamos catalogar como muy sencillas. Los colores se eligen con combinaciones de funciones aleatorias, como ruidos muy correlados (movimientos fraccionarios brownianos, que suena muy sofisticado pero es muy sencillo de comprender), y sobre todo funciones trigonométricas simples. Los movimientos, a su vez, son resultado de funciones de deformación (o aplicaciones, o “mapeos”) sobre el plano básicas, y debidamente optimizadas (simplificadas) para poder ser evaluadas a gran velocidad en un ordenador de sobremesa de potencia baja.

Como de costumbre, resulta mucho más sencillo experimentar uno mismo las cosas que intentar imaginárselas. Por eso quiero desarrollar este apartado de forma experimental mediante ejemplos en vivo, para lo cual necesito que el lector visite la siguiente página web y se descargue el programa de software *Plane Deformations*. Ahora, veamos como es posible jugar con las matemáticas.

Arranque el programa de ordenador. Lo primero que verá es una pantalla como la que se muestra a la derecha de este texto, y sobre ella una ventana de ayuda que puede cerrar. Para comenzar, mueva las barras de desplazamiento con nombres "*u offset speed*" y "*v offset speed*" ligeramente hacia la izquierda hasta su posición central, hasta que la imagen de baldosas se detenga. Fíjese en los tres cuadros de texto de la parte inferior de la ventana del programa, los que llevan por título "u=", "v=" y "w=" Es ahí donde vamos a jugar la mayor parte del tiempo. Para empezar, reemplace la "x" con "2*x" (un asterisco entre el "2" y la "x") y presione el botón que lleva el texto "Apply".



Verá que la imagen de baldosas se ha aplastado (comprimido) horizontalmente. Compruebe que cambiando la actual "v=y" por "v=3*y" aplasta la imagen verticalmente. Como ve, u y v son dos funciones que mapean cada punto del plano a otro. Es decir, el cuadrado central del programa muestra el plano euclídeo, con su origen de coordenadas $(0,0)$ centrado en la pantalla, y las funciones u y v traen a cada punto $p(x,y)$ de este plano un punto $p'(x',y')$ de otro plano donde reside la imagen de ladrillos. Es decir, estamos deformando el plano de ladrillos mediante las funciones "x'=$u(x,y)$" y "y'=$v(x,y)$". Compruébelo escribiendo "u=2*x+y", o cualquier otra expresión similar.

A continuación mueva las barras de desplazamiento "*u offset speed*" y "*v offset speed*" hacia la derecha o la izquierda, no importa cómo. Comprobará que la imagen de baldosas se desplaza de forma lineal (velocidad constante) sobre el plano pre-deformado. También puede mostrar los ejes de coordenadas pinchando sobre el recuadro "*axes*" arriba a la derecha.

A partir de este punto, todo es experimentación (poco a poco llegará la comprensión del funcionamiento, pero eso es posterior). Por ejemplo, reescriba "u=x" y "v=y" para volver al plano que teníamos originalmente sin deformar, y escriba "u=$abs(x)$", donde $abs()$ significa "valor absoluto". Compruebe que se obtiene una imagen en espejo. Efectivamente, el valor absoluto de x provoca, en términos vulgares, que "todo lo que está a la izquierda (en la parte de las x negativas) es igual a lo que está a la derecha (x positivas)". Pruebe ahora "u = x*x" es decir, x al cuadrado, que también se puede rescribir como "u = $xf(x)$=x^2", que es una parábola centrada en el origen, es una función "par". Además, analizando un poco más la imagen vemos que los ladrillos son más grandes en el centro de la pantalla y más pequeños en los bordes. Esto es porque la parábola tiene una derivada mayor lejos del origen, y por tanto "comprime" más el plano de las baldosas. Podríamos pensar en la derivada de una función como en el grado de "compresión" o "expansión" que produce dicha función en cada punto. En cualquier caso, dada una función de deformación es muy fácil, por tanto, predecir (o mejor aún, planear) dónde los ladrillos serán más grandes o más pequeños sin más que mirar la derivada. En cualquier caso, experimentemos ahora con polinomios de grado mayor, por ejemplo "u=x^2–y+3*x*y" y "v=x*y^3–x+y".

Las funciones pares producen imágenes en espejo.

Probemos ahora "u=$log(x)$", "v=y", donde "log" significa logaritmo (natural). Otra vez, la parte central de la pantalla tiene ladrillos más pequeños que la parte derecha, debido a la derivada del logaritmo que es cada vez más pequeña (visualice la gráfica de un logaritmo real). Fijémonos ahora en la parte izquierda de la pantalla, que se ve "fea". Eso es porque en esa zona, donde las x son negativas, el logaritmo no existe (toma valores complejos). El ordenador es una gran calculadora, y como tal sólo sabe manejar números. Cuando se trata de valores que "no existen" o infinitos, los resultados son siempre raros, o feos, como podemos ver en este caso. Veamos otro ejemplo: "u=x" y "v=1/y". Ahora, en la parte central de la pantalla hay una franja horizontal que se ve "fea" otra vez. Se debe, claro está, al infinito que

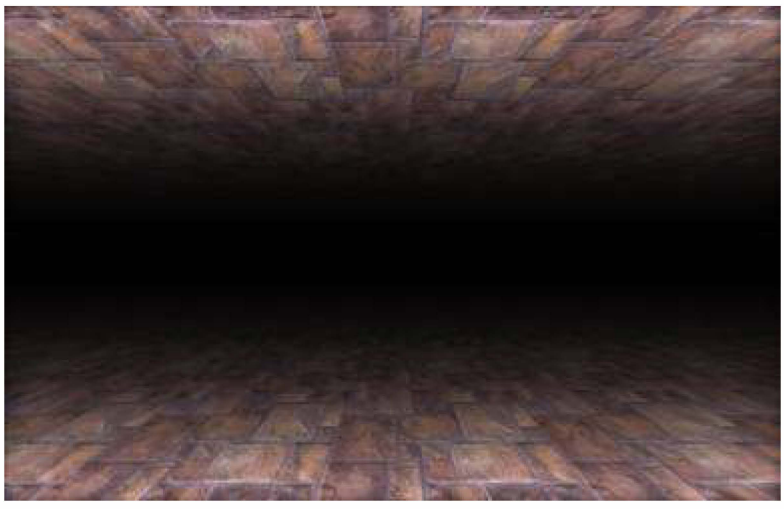
estamos creando cuando la y vale cero o se le acerca mucho, es decir, cuando nos acercamos (por arriba o por abajo) al centro de la pantalla.

Sigamos con este ejemplo y experimentemos un poco. Cambiemos sólo la función u, haciendo "$u=x/y$" y "$v=1/y$". Tenemos aquí una deformación muy interesante, sobre todo si ahora alteramos levemente las barras de desplazamiento "*u offset speed*" y "*v offset speed*". Parece que estuviésemos inmersos en un espacio tridimensional con dos planos infinitos paralelos el uno al otro, justo encima y debajo de nuestra cabeza. Pero notamos que los movimientos de estos planos virtuales tienen direcciones contrarias: arriba se mueve en una dirección, y en la parte de debajo de la pantalla en la otra. Concluimos que el efecto debe estar relacionado con el signo de la y. Así que lo podríamos intentar arreglar anulando el efecto del signo de la y en las ecuaciones; eso debería funcionar. Usemos, por tanto, la herramienta que aprendimos hace unos momentos para hacer imágenes en espejo (el valor absoluto): cambiemos todas las "y" por "$abs(y)$" para obtener "$u=x/abs(y)$" y "$v=1/abs(y)$". Pulsamos "Apply", y efectivamente tenemos lo que queríamos. Ahora sólo nos queda mover la barra "*u offset speed*" hasta su posición central y la barra "*v offset speed*" a la derecha del todo para tener un viaje virtual por un espacio tridimensional. Pero aún podemos hacer que la deformación sea más convincente mediante la tercera función que el programa nos ofrece y que aún no hemos utilizado.

Bajo los cuadros de texto para la u y la v, tenemos un tercero para la w, que por defecto tiene el valor "1.0". Este programa de ordenador utiliza la función w para variar el brillo de la imagen. Como sabemos, para el ordenador todos los datos son sólo números, desde texto hasta música, el brillo de cada punto de la pantalla incluido. Por ejemplo, un punto de pantalla con un valor de brillo 0.0 equivale a brillo mínimo (el punto se vería absolutamente negro), mientras que un brillo con valor 1.0 equivale a brillo normal, que es el que tenemos ahora en el programa por defecto. Compruébelo cambiando a "$w=0.6$" ó "$w=1.3$" y viendo la reacción en la imagen. Bien, como supondrá ahora nada nos impide utilizar una función matemática para el brillo en lugar de un valor constante. Por ejemplo, introduzca "$w=0.5+0.5*x$", y compruebe que efectivamente la ecuación de la recta que acabamos de introducir explica el degradado de brillo que observamos en la pantalla. Evidentemente, la recta "$w=x+y$" hace lo propio.

Ahora, ajustemos la w para que el brillo concuerde con la deformación de baldosas que teníamos ("$u=x/abs(y)$", "$v=1/abs(y)$". Por ejemplo, un brillo "$w=abs(y)$" acompaña bastante bien a la deformación. Supongamos ahora que quisiéramos reducir la cantidad de área oscura que acabamos de introducir, pero deseamos seguir manteniendo la zona central de la pantalla negra. Lo que podemos hacer es utilizar para ello es una raíz cuadrada. Recuerde que la función brillo ahora mismo toma valores de cero a uno según nos movemos desde la parte central de la pantalla hacia los extremos superior e inferior. Queremos que la zona negra siga siendo negra, y que las zonas extremas ($abs(y)=1$) sigan teniendo brillo 1. Pero queremos que las zonas cercanas a la zona negra, sean menos negras. Es decir, algo que mapee cero a cero, uno a uno, y amplifique las zonas intermedias. Ahora visualice la curva de una raíz cuadrada en el eje real, entre las abscisas 0 y 1, y verá que es justo lo que necesitamos: pasa por los puntos $(0,0)$ y $(1,1)$ al tiempo que hace que los valores cercanos a cero "crezcan". Probamos por tanto "$w=sqrt(abs(y))$" (donde "*sqrt*" significa raíz cuadrada), y vemos que hemos obtenido lo que queríamos.

En resumen, hemos llegado a este "efecto" de planos horizontales infinitos mediante pura experimentación (al menos inicialmente) de fórmulas en principio arbitrarias. Evidentemente, el hecho de que "veamos" dos planos paralelos horizontales infinitos se puede explicar muy sencillamente mediante conceptos de geometría tridimensional y proyección (o intersección de planos y rectas). En cualquier caso, sea posible o no explicar o diseñar un efecto visual dado, las demos suelen contener una combinación de ambas opciones. Gran parte de los efectos son conseguidos por pura experimentación, mientras que otros son resultado de búsquedas conscientes de ciertos objetivos. Con la experimentación sucesiva y el tiempo suficiente, la balanza suele inevitablemente inclinarse hacia el diseño consciente de fórmulas.



Las funciones "$u=x/|y|$, $v=1/|y|$, $w=|y|$" producen un interesante efecto visual.

Jugando sobre el plano, en coordenadas polares

Cambiemos rápidamente a ecuaciones polares, que tradicionalmente han sido mucho más utilizadas en la demoscene. Recordemos que las coordenadas polares de un punto en el plano son los valores de la distancia el punto al origen de coordenadas, y el ángulo que forma la recta que une el punto y este origen de coordenadas con el eje horizontal. Para hacer uso de estas coordenadas, el programa permite el uso las letras "r" (para radio, o distancia) y "a" (para el ángulo). Por ejemplo, probemos "$u=r$", "$v=a$" y "$w=1$". Veremos que hemos creado una especie de "agujero" que se traga el plano de baldosas. Mueva la barra "*u offset speed*" de izquierda a derecha para crear un efecto muy curioso.

Tal vez habrá notado que en la parte izquierda del plano hay una discontinuidad en la imagen de baldosas, sobre el eje horizontal. Se debe básicamente a que en esa zona la función v es discontinua porque el ángulo pasa de valer $-\pi$ a valer π (o, visto de otro modo, la función arco-tangente se ramifica en esta semi-recta). Existe un truco sencillo para solventarlo, y consiste en dos pasos. Primero, utilizar un patrón de baldosas que sea periódico (que se repita cada cierto espacio); y segundo, ajustar dicho periodo (o separación de la repetición) a un múltiplo de 2π en este caso. Por razones de eficiencia computacional, el programa utiliza un periodo de repetición de baldosas de 1, de modo que lo que podemos hacer en cambio es ajustar la discontinuidad de nuestra función v a un múltiplo de este periodo, es decir, a cualquier número entero. Lo intentaremos con 3, que es el entero más cercano a π: "$v=3*a/pi$" (si todavía ve la discontinuidad se trata ahora de detalles técnicos y no matemáticos, y los puede resolver cambiando ligeramente el tamaño de la ventana del programa).

Sigamos jugando con la deformación. Por ejemplo, alteremos v según la distancia al centro de la pantalla: "$v=3*a/pi + 2*r$". Vemos que cuanto más nos alejamos del centro de la pantalla, más se retuercen las baldosas. Cambiemos este comportamiento utilizando un coseno, por ejemplo: "$v=3*a/pi + cos(r)$". Podemos acentuar más el efecto del coseno haciendo "$v=3*a/pi + cos(3*r)$". Sabemos que el coseno sube y baja sin parar, así que lo que estamos haciendo realmente es que la cantidad de "torcimiento", que antes era creciente (lineal), ahora oscile según nos alejamos del centro de la pantalla: primero retorcemos las baldosas mucho, luego poco, luego mucho, etc.

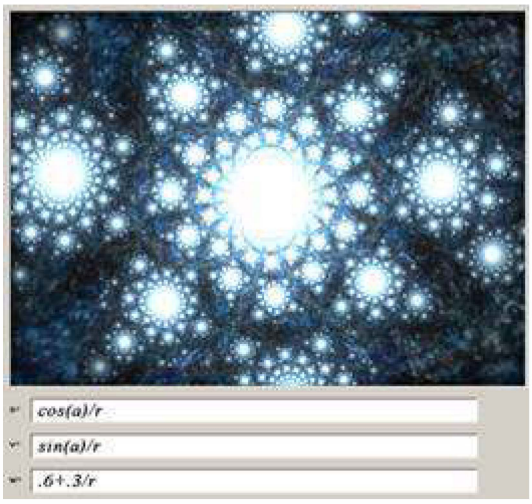
Volvamos a "*u*=*r*", "*v*=3**a*/*pi*" y "*w*=1". Veamos otro truco de geometría tridimensional elemental: cambiemos la función *u* tan sólo un poco: "*u*=1/*r*". Tenemos un cilindro. Y ahora que sabemos manejar el brillo *w*, sólo nos queda hacer "*w*=*r*" o, incluso, "*w*=0.5 + 0.43/*r*" para llevar al espectador a través de un túnel infinito. Recordando que la "*r*" no es sino la distancia euclídea al centro de coordenadas, o, en otras palabras, (*x*^2+*y*^2)^(1/2), es casi inevitable caer en la tentación de generalizar esta expresión y probar, por ejemplo, "*u*=1/(*x*^32+*y*^32)^(1/32)" en vez de "*u*=1/*r*". Hagámoslo, y ajustemos el brillo también: "*u*=1/(*x*^32+*y*^32)^(1/32)". Llevando la barra "*v offset speed*" a su posición central veremos que ahora viajamos por un túnel con sección casi cuadrada.

Este tipo de deformaciones con forma de túneles son uno de recursos más antiguos usados en la demoscene y fueron muy populares a comienzos de los años noventa, ya que son rápidos de calcular incluso en los ordenadores de aquella época, y dan resultados muy espectaculares.

Jugando sobre el plano, realimentación

Para acabar con esta sección de deformaciones del plano, veamos otro de los recursos más viejos de la demoscene: la realimentación de efectos visuales. Para ello volvamos brevemente a la deformación identidad "$u=x$", "$v=y$", "$w=1$". Localice la barra "*feedback coverage*" y desplácela hacia la derecha hasta aproximadamente su posición central. Mueva asimismo la barra inmediatamente debajo, con nombre "*feedback strength*". Como puede comprobar, en la imagen han aparecido cierto número de recuadros que muestran cada uno una copia de la imagen completa. Siendo más precisos, muestran la imagen completa tal y como era "hace un instante", de ahí el nombre de *realimentación*. Y las dos barras mencionadas nos ayudarán a controlar esta realimentación: con la primera ajustamos el área de los recuadros realimentados, y con la segunda su ganancia (brillo relativo).

Ahora, cambiemos las funciones: "$u=r*cos(a)$", "$u=r*sin(a)$". Comprobamos que la imagen no se ha alterado, ya que como sabemos no hemos hecho más que rescribir "$u=x$" y "$v=y$" en su forma polar. Sin embargo, ahora podemos hacer cosas interesantes que antes no podíamos, como cambiar el argumento del seno y del coseno de "a" a "$a+2*r$". O podemos alterar el módulo de "r" a "$1/r$": "$u=cos(a)/r$", "$v=sin(a)/r$". De hecho, si cada punto del plano representara un número complejo, esta sería la transformación $z \to 1/z$, que es una especie de inversión del plano en la que todo lo que está lejos del origen (r grandes) se acerca (se hacen pequeñas) y viceversa, algo que puede comprobar visualmente deshabilitando la realimentación.



En cualquier caso, la realimentación le da un aire interesante a la imagen. Ahora, juguemos con el brillo un poco, por ejemplo con "$w=.5 + .3/r$" para hacer la parte central de la imagen ($r=0$) muy brillante, y también, por tanto, el centro de cada una de las sub-imágenes realimentadas (dando una imagen semejante a un fractal, aunque estrictamente no lo es, porque recordemos que en nuestro caso cada sub-imagen no es copia de la imagen entera que vemos, sino de una imagen que vimos en el pasado). Por último, ajuste la expresión del brillo y los parámetros de realimentación hasta obtener la imagen que más le guste.

Ahora piense que todos estos parámetros (*"u/v offset speed*" y "*feedback strengh*") no fueran constantes, sino que fueran funciones (matemáticas) del tiempo, y que diseñásemos esas funciones para ir al ritmo de cierta música. Ya empieza a ver cómo son las bases para la creación de una demo...

Un poco más a fondo

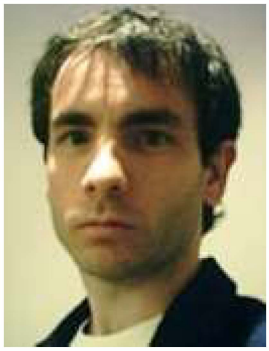
... Bueno, no exactamente (puede ya cerrar el programa de deformaciones). Las cosas son un poco más complicadas, no a nivel matemático sino de programación. Por ejemplo, no hemos entrado en detalles, pero posiblemente sepa que la imagen de ladrillos que hemos utilizado es una imagen discreta, no continua, compuesta por un número finito de puntos (píxeles). Sin embargo, nuestras funciones de mapeado u y v arrojan números reales. Y por tanto surge la cuestión de qué hacer si u y v nos dicen que accedamos al punto/píxel $(13.2, 97.6)$ de la imagen. Evidentemente podríamos simplemente acceder al punto más cercano, por ejemplo $(13, 98)$... Por desgracia, esto suele dar lugar a una pérdida de calidad visual considerable. Es necesario, por tanto, mapear las coordenadas x' e y' del plano no a un punto de la imagen, sino a un color, y nos ayudaremos de la imagen para obtenerlo. Para ello se suele utilizar algún tipo de interpolación, como bilineal o bicúbica (o algún tipo de spline). Pero todo esto no lo hemos explicado en el artículo: es una ciencia en sí misma.

Si fuéramos a más bajo nivel, tendíamos que explicar además que la imagen de baldosas que hemos utilizado tiene dimensiones de 512×512 puntos, un número no arbitrario sino meditadamente seleccionado. De hecho, 512 es una potencia de 2 (es decir, un número binario "redondo"), lo cual simplifica ciertos cálculos en el ordenador (del mismo modo que para los humanos, números redondos como 100 ó 1000 hacen los cálculos más sencillos que otros como 937 ó 1031). Y cálculos más sencillos ayudan a que todo el programa de deformaciones anterior funcione en tiempo real.

Finalizando

Antes de acabar el artículo me permito pasarles el siguiente enlace de Internet donde podrán encontrar una pequeña selección de demos que recomiendo disfruten, con estilos variados. También recomiendo echar un vistazo a las deformaciones pregrabadas del programa *Plane Deformations* que son accesibles a través del menú (ir a la carpeta "*data*").

Lamentablemente, no existe demasiada información sobre la demoscene, y mucho menos páginas u organizaciones oficiales que la soporten; la demoscene es simplemente un grupo de jóvenes entusiastas que se mueven por pura pasión. Es difícil, por tanto, pasarles enlaces a sitios de Internet con más información; de momento, sólo existe una página en castellano sobre el tema que está aún en construcción y contiene información demasiado técnica. Esperamos poder solventar esto en breve.



Sobre el autor

Íñigo Quílez es ingeniero de telecomunicaciones por la Universidad del País Vasco / EHU, y actualmente trabaja como ingeniero de software en la empresa de realidad virtual VRcontext (Bruselas, Bélgica), diseñando algoritmos de visualización 3D. Su pasión es el uso creativo de las matemáticas para hacer algo bello, y la divulgación.

matematicalia
revista digital de divulgación matemática