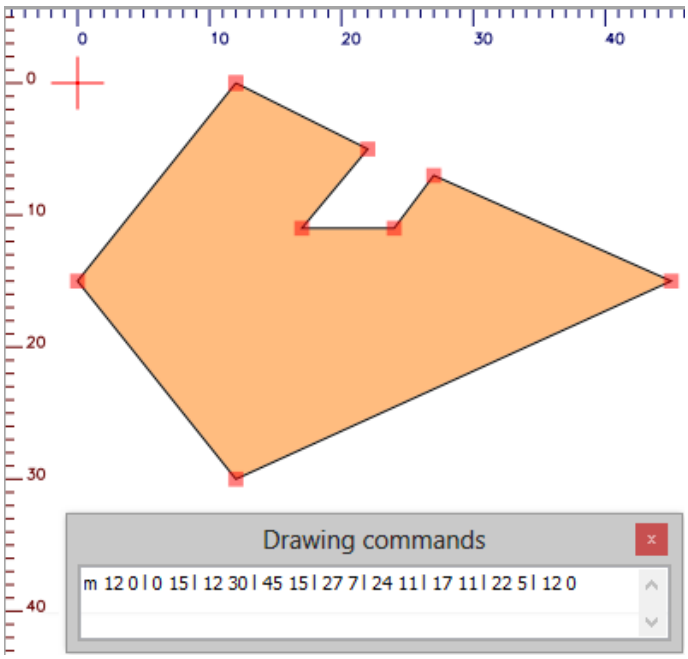
Kara Effector 3.2:

En el **Tomo XVI** seguimos con la funciones de la Librería **shape**, pero con la certeza que en este **Tomo** no terminará la misma, ya que es muy extensa y completa con el fin que tengamos a nuestra plena disposición la mayor cantidad de herramientas posibles a la hora de desarrollar nuestros proyectos.

Librería Shape [KE]:

Para los siguientes ejemplos en las definiciones de las funciones, usaremos esta simple **shape** que de 45 X 30 px:



Y como ya es costumbre, declaramos una variable con nuestra **shape** (el nombre es a gusto de cada uno):

Variables:

```
mi_shape = "m 12 0 l 0 15 l 12 30 l 45 15 l 27 7 l 24 11
            l 17 11 l 22 5 l 12 0 "
```

shape.array(shape, loops, A_mode, Dxy): hace un Arreglo o Matriz (duplicaciones) de la **shape**, una cierta cantidad de veces (**loops**), con diversas características y modalidades dependiendo de los otros dos parámetros (**A_mode** y **Dxy**).

Esta función tiene tres modalidades distintas, y cada una tiene una serie de opciones que veremos a continuación en los siguientes ejemplos:

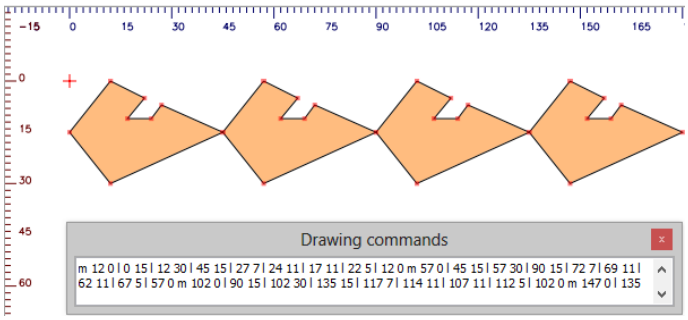
Modo Lineal

- **Ejemplo 1.** Ingresamos la **shape**, el número de repeticiones, que para este ejemplo es 4, y no se pone ni **A_mode** ni **Dxy**:

Return [fx]:

```
shape.array( mi_shape, 4 )
```

Entonces la **shape** se duplicará una a la derecha de la otra, cuatro veces, en forma lineal. El duplicar la **shape** una a la derecha de la otra es equivalente a un Arreglo Lineal con un ángulo de 0°:

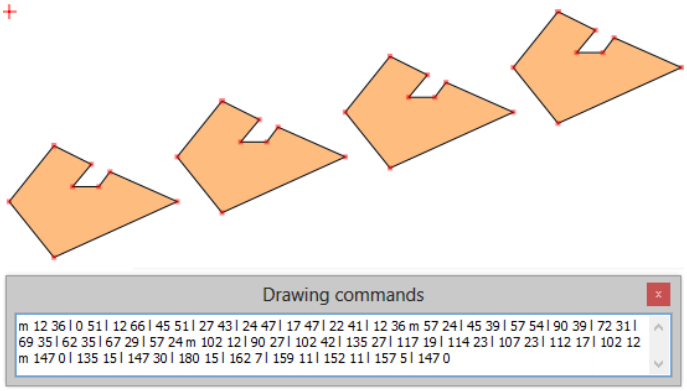


- **Ejemplo 2.** Aparte de la **shape** y la cantidad de repeticiones, el parámetro **A_mode** es 15, que es equivalente a un ángulo de 15°.
  Cuando **A_mode** es un valor numérico, la función asume que dicho valor es el ángulo que tendrá el Arreglo Lineal de la shape resultante:

Return [fx]:

```
shape.array( mi_shape, 4, 15 )
```

El Arreglo es Lineal y con un ángulo de 15°:

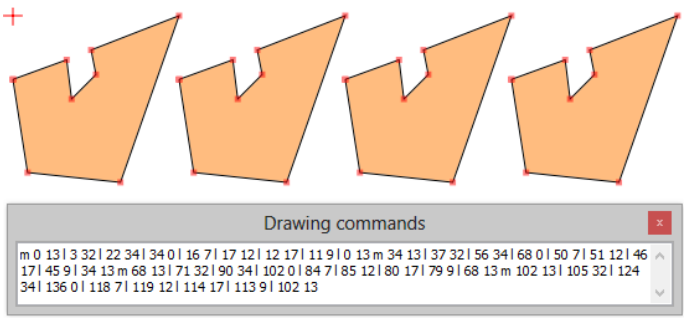


- **Ejemplo 3. A_mode** ahora es una **tabla** con dos valores numéricos, el primero indica el ángulo del arreglo lineal y el segundo, la rotación respecto al centro de la **shape**:

Return [fx]:

```
shape.array( mi_shape, 4, {0, 45} )
```

Entonces el ángulo del arreglo es 0 y la **shape**, antes de ser duplicada, se rotó un ángulo de 45°:

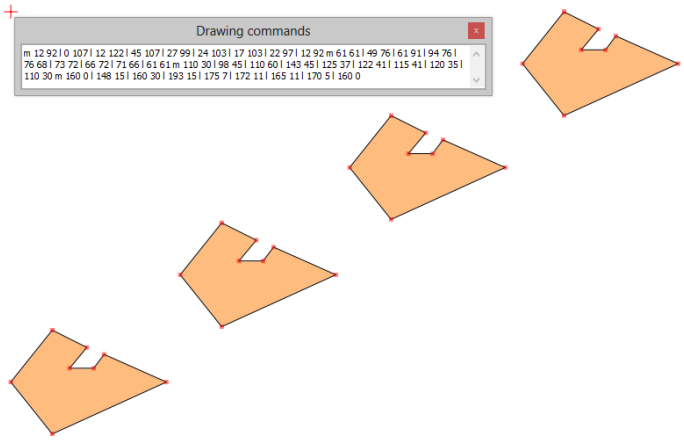


- **Ejemplo 4**. Incluimos al parámetro **Dxy** como valor numérico, que hace referencia a la distancia en px que separará a la **shape** dentro del arreglo:
  - loops : 4
  - Ángulo del Arreglo: 32°
  - Distancia Separadora: 5 px

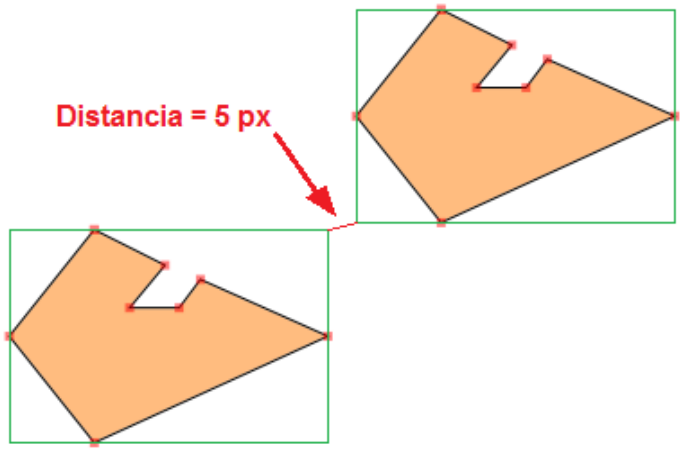
Return [fx]:

```
shape.array( mi_shape, 4, 32 , 5)
```

La distancia entre las Shapes del Arreglo ahora ya no es cero, que es su valor por default, sino 5 px:



En la anterior imagen queda la sensación de que las Shapes están separadas por una distancia mayor a 5 px, pero es porque la forma de la shape no es totalmente rectangular. Si tomamos dos de ellas y las enmarcamos en rectángulos, su pueden ver los 5 px de separación:



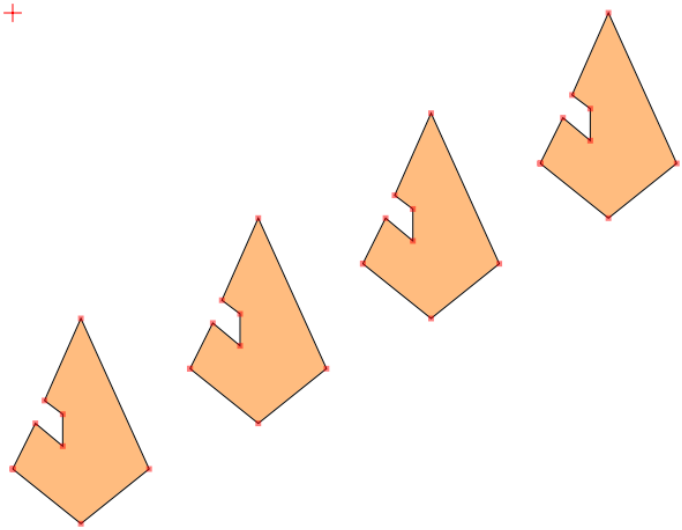
La distancia de separación también puede ser un valor numérico negativo, y lo que hará que el Arreglo Lineal quede más compacto y se acerquen tanto, una shape a la otra, hasta que se superpongan, si eso quisiéramos.

- **Ejemplo 5.** Es una combinación de los ejemplos 3 y 4. Ahora podemos decidir la separación, el ángulo del Arreglo y el ángulo de rotación:

Return [fx]:

```
shape.array( mi_shape, 4, {30, 90} , 10)
```

- loops: 4
- Ángulo del Arreglo: 30°
- Ángulo de Rotación de la **shape**: 90°
- Distancia Separadora: 10 px



Los anteriores ejemplos nos muestran las cinco opciones de Arreglos Lineales que podemos hacer con la función. El **Modo Lineal** es la forma más simple de usar esta función, pero a continuación veremos el próximo modo de uso y sus diversas opciones:

Modo Radial

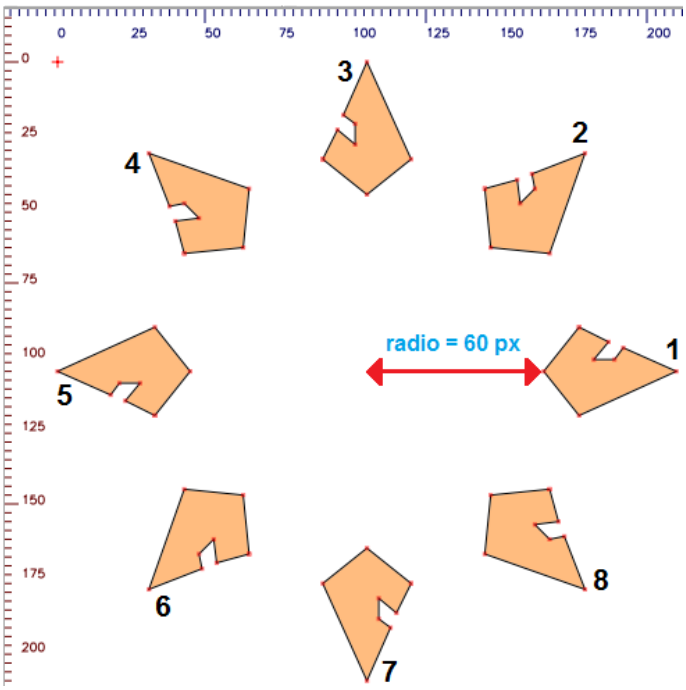
- **Ejemplo 1.** Ingresamos la **shape**, el número de repeticiones del Arreglo, en **A_mode** escribimos entre comillas la palabra "**radial**". **Dxy** equivale al radio del arco:

Return [fx]:

```
shape.array( mi_shape, 8, "radial", 60 )
```

Vemos la **shape** repetida ocho veces. El ángulo del arco es 360° por default, y es por eso que la **shape** se repite en un Arreglo Radial, equidistantemente en esos 360°:

- loops: 8
- Modo: “radial”
- Radio: 60 px



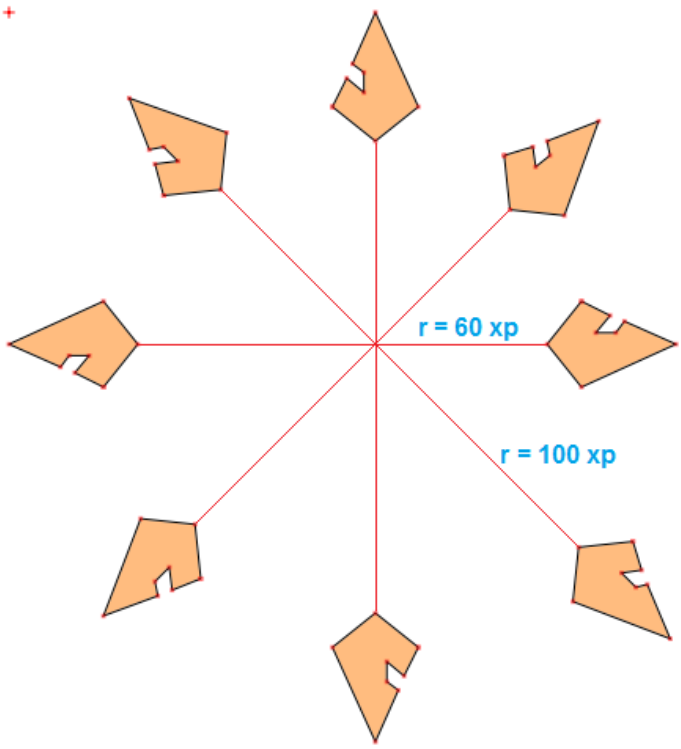
El radio es constante para cada una de las repeticiones del Arreglo (60 px), y cada una de las Shapes está rotada de tal manera que quedan orientadas en dirección del centro imaginario del Arreglo.

- **Ejemplo 2.** Ingresamos la **shape**, el número de repeticiones del Arreglo, en **A_mode** escribimos entre comillas la palabra “**radial**”. Pero ahora **Dxy** es una **tabla** que contiene dos valores numéricos, el primero equivale al radio inicial en pixeles del Arreglo y el segundo al radio final:
  - loops: 8
  - Modo: “radial”
  - Radio Inical: 60 px
  - Radio Final: 100 px

Return [fx]:

```
shape.array( mi_shape, 8, "radial", {60, 100} )
```

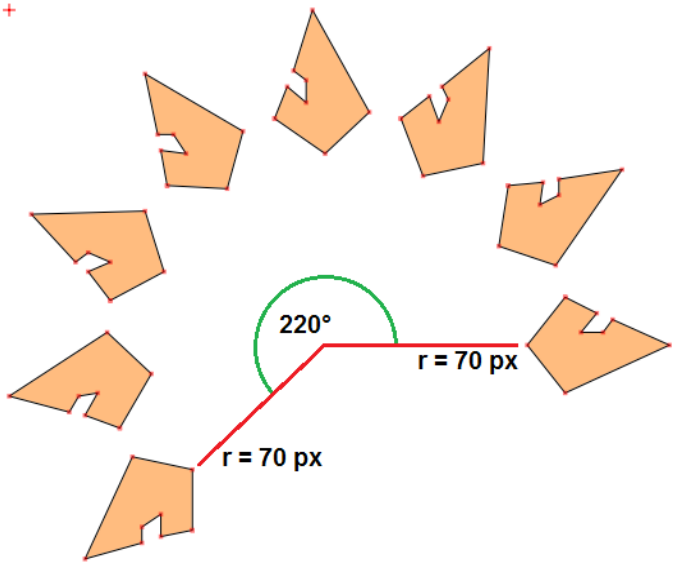
Los radios del Arreglo van aumentando progresivamente desde el Radio Inicial (60 px) hasta el Radio Final (100 px):



- **Ejemplo 3.** Agregamos un tercer valor en la **tabla Dxy**, equivalente al ángulo del arco del Arreglo, que por default era 360°:

Return [fx]:

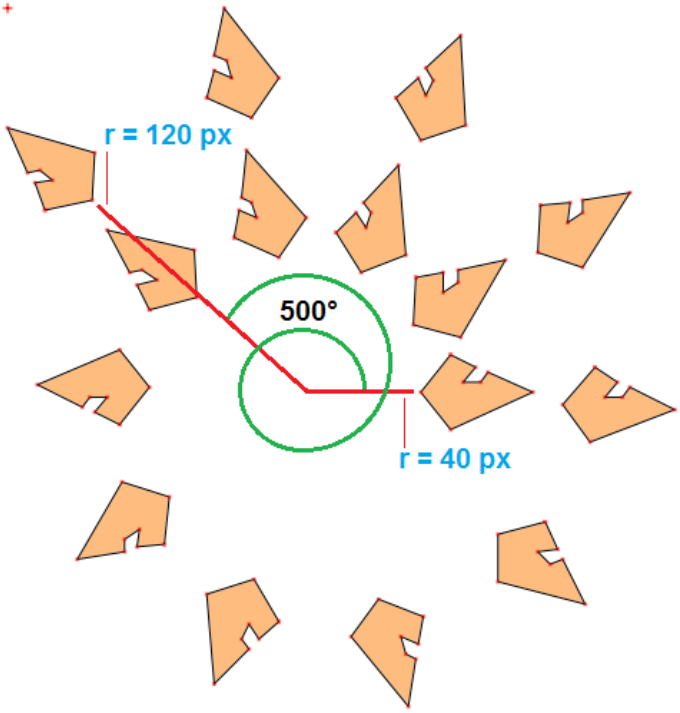
```
shape.array( mi_shape, 8, "radial", {70, 70, 220} )
```



- **Ejemplo 3.1.** Aumentamos el valor del ángulo del Arreglo de tal manera que exceda los 360° de la circunferencia y, ponemos al Radio Inicial y al Radio Final con diferentes valores para evitar que las Shapes se superpongan:

```
Return [fx]:
shape.array( mi_shape, 15, "radial", {40, 120, 500} )
```

  - loops: 15
  - Modo: “radial”
  - Radio Inical: 40 px
  - Radio Final: 120 px
  - Ángulo del Arco: 500°

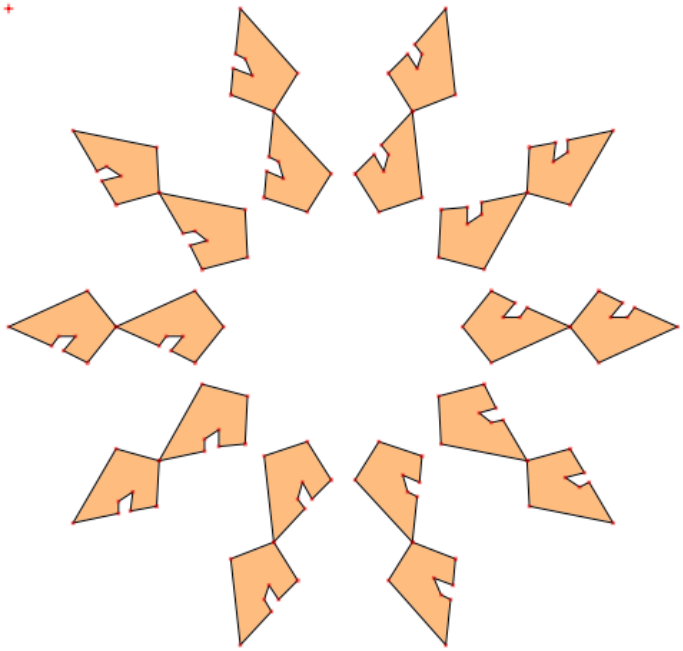


- **Ejemplo 4.** El parámetro **loops** es una **tabla** con dos valores numéricos, el primero indica las veces que se repite la **shape** en el Arreglo Radial, y el segundo indica la veces que se repite el Arreglo de forma y tamaño mayormente progresivo:

```
Return [fx]:
shape.array( mi_shape, {10, 2}, "radial", 50 )
```

  - loops del Arreglo: 10
  - Repeticiones: 2
  - loops Total: 10 X 2 = 20
  - Modo: “radial”
  - Radio Interior del primer Arreglo: 50 px

Al usar la función de esta forma, la separación entre cada uno de los Arreglos Radiales es por default 0, es decir que el radio interior de cada Arreglo Radial será exactamente del mismo tamaño en pixeles del radio exterior del Arreglo inmediatamente anterior:



La **shape** resultante es cada vez más compleja y resultaría muy laborioso dibujarla manualmente en el **AssDraw3**, además del hecho de no poder hacerlo con tanta precisión y velocidad. Del anterior ejemplo resulta una **shape** muy interesante para hacer un Efecto:

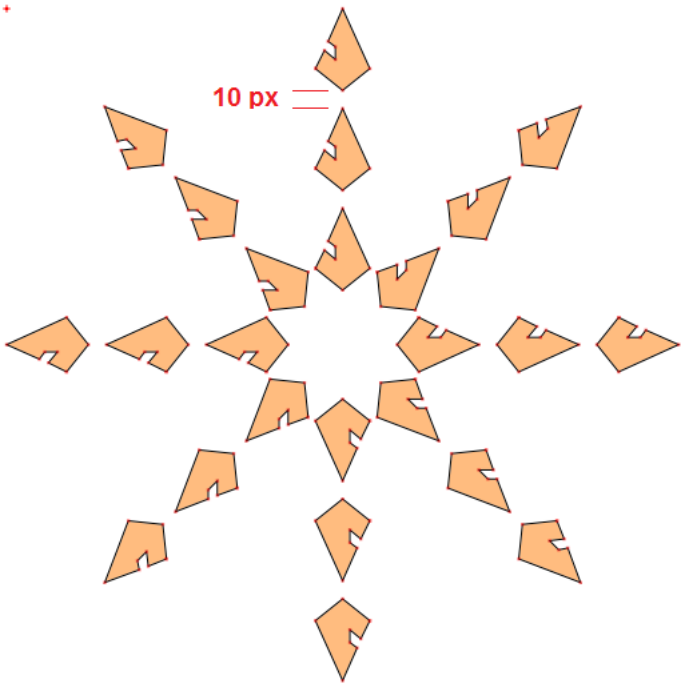
Con la **shape** del **Ejemplo 1**:

- **Ejemplo 5.** El parámetro **Dxy** es ahora una **tabla** con dos valores numéricos, el primero equivale al radio interior del primer Arreglo y el segundo, a la distancia en pixeles que separará a cada uno de los Arreglos:

Return [fx]:

```
shape.array( mi_shape, {8, 3}, "radial", {30, 10} )
```

  - loops del Arreglo: 8
  - Repeticiones: 3
  - loops Total: 8 X 3 = 24
  - Modo: “radial”
  - Radio Interior del primer Arreglo: 30 px
  - Distancia Separadora entre Arreglos: 10 px



Para los siguientes ejemplos convertiremos a la variable **mi_shape** en una **tabla**, en donde el primer elemento será la shape que inicialmente ya teníamos y el segundo será un círculo de 24 px de diámetro:

Variables:

```
mi_shape = { "m 12 0 l 0 15 l 12 30 l 45 15 l 27 7 l 24 11
l 17 11 l 22 5 l 12 0 ", shape.size(shape.circle, 24) }
```

Círculo de 24 px

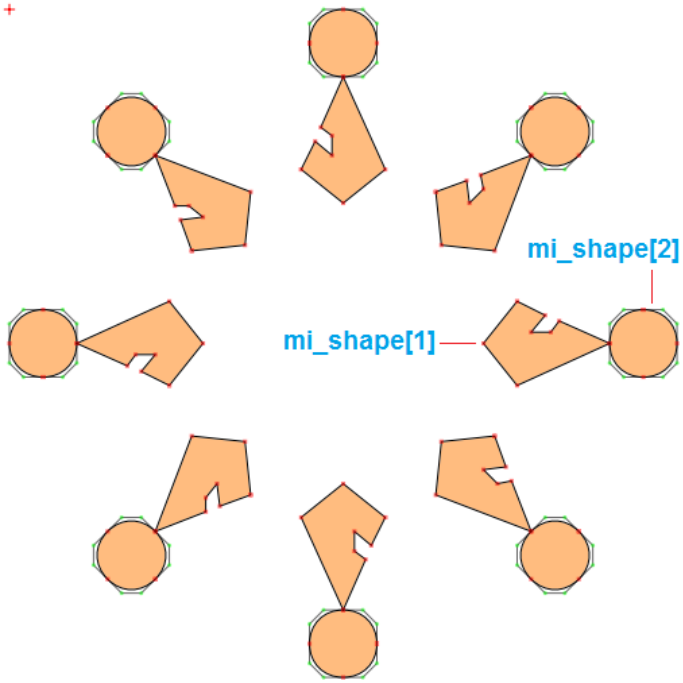
La **tabla mi_shape** de ser completamente de Shapes para poder ser usada dentro la función, así que pueden usar las de estos ejemplos o las que ustedes quieran. La cantidad de Shapes de la **tabla** es ilimitada.

- **Ejemplo 6. mi_shape** es una **tabla** de Shapes:
  - loops del Arreglo: 8
  - Repeticiones: 2
  - loops Total: 8 X 2 = 16
  - Modo: “radial”
  - Radio Interior del primer Arreglo: 50 px

Return [fx]:

```
shape.array( mi_shape, {8, 2}, "radial", 50 )
```

La función toma a la primera **shape** de la **tabla** y la repite en el primer Arreglo, luego toma la segunda para el segundo Arreglo, y así de manera sucesiva para el caso en que la **tabla mi_shape** tenga todavía más elementos.



En la anterior imagen notamos cómo el primer Arreglo está hecho con las repeticiones de **mi_shape[1]** (o sea, el primer elemento de la **tabla mi_shape**) y para el segundo Arreglo se usó **mi_shape[2].**

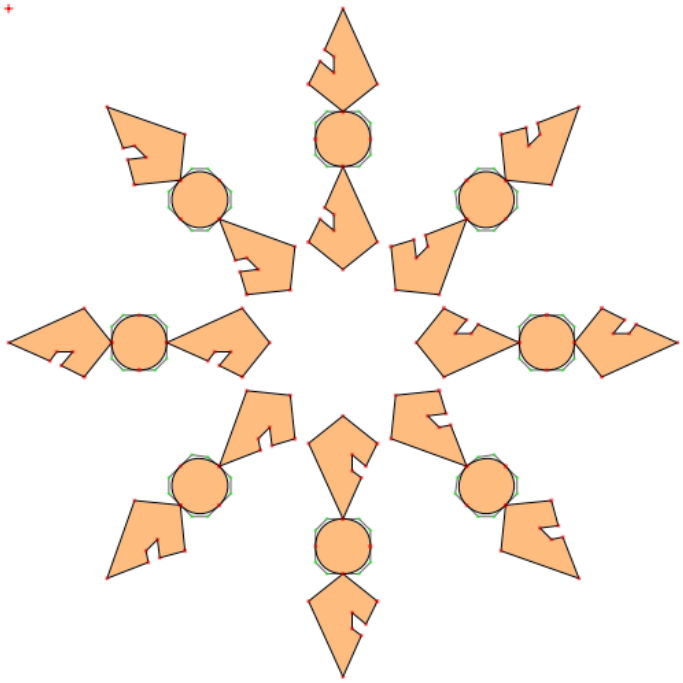
La distancia que separa un Arreglo respecto a otro es 0 px, por default, pero también la podemos modificar.

- **Ejemplo 6.1. mi_shape** es una **tabla** de Shapes:
  - loops del Arreglo: 8
  - Repeticiones: 3
  - loops Total: 8 X 3 = 24
  - Modo: “radial”
  - Radio Interior del primer Arreglo: 32 px

Return [fx]:

```
shape.array( mi_shape, {8, 3}, "radial", 32 )
```

Este ejemplo está hecho para mostrar que no importa que la cantidad de repeticiones de los Arreglos asignada en la tabla **loops** (o sea 3) exceda a la cantidad total de Shapes en la **tabla mi_shape** (**#mi_shape = 2**). La función tomará para el tercer Arreglo nuevamente a la primera **shape** de la **tabla**:

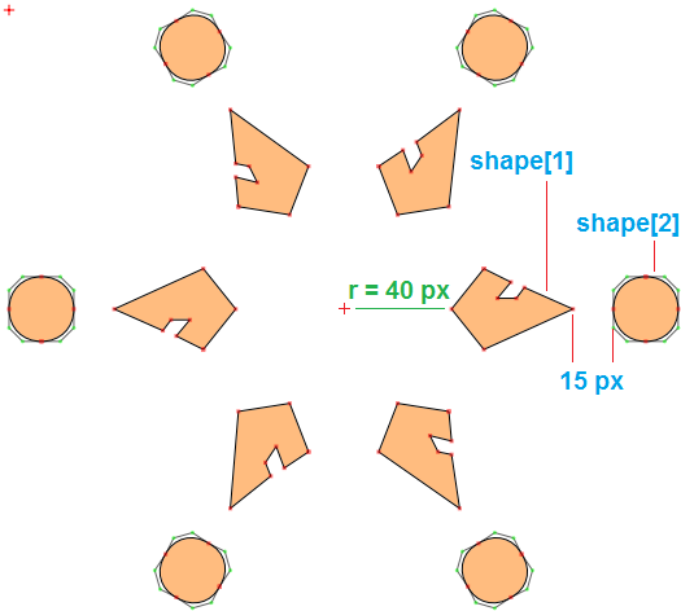


- **Ejemplo 7. mi_shape** es una **tabla** de Shapes y en **Dxy**, en forma de **tabla**, modificamos la distancia que separará a cada uno de los Arreglos:

Return [fx]:

```
shape.array( mi_shape, {6, 2}, "radial", {40,15} )
```

- loops del Arreglo: 6
- Repeticiones: 2
- loops Total: 6 X 2 = 12
- Modo: “radial”
- Radio Interior del primer Arreglo: 40 px
- Distancia Separadora: 15 px



Acá es el final del **Tomo XVI**, pero la función **shape.array** aún tiene muchos más secretos que revelarnos y quedan muchos ejemplos para poner en práctica. Literalmente las opciones son infinitas.

En el **Tomo XVII** continuaremos con el tercer **Modo** de uso de la función **shape.array** y con las demás funciones de la Librería **shape**. Intenten poner en práctica todos los ejemplos vistos en este **Tomo** y no olviden descargar la última actualización disponible del **Kara Effector 3.2** y visitarnos en el **Blog Oficial**, lo mismo que en los canales de **YouTube** para descargar los nuevos Efectos o dejar algún comentario, exponer alguna duda o hacer alguna sugerencia. Pueden visitarnos y dejar su comentario en nuestra página de **Facebook**:

www.facebook.com/karaeffector