

Trabajo Practico N° 1

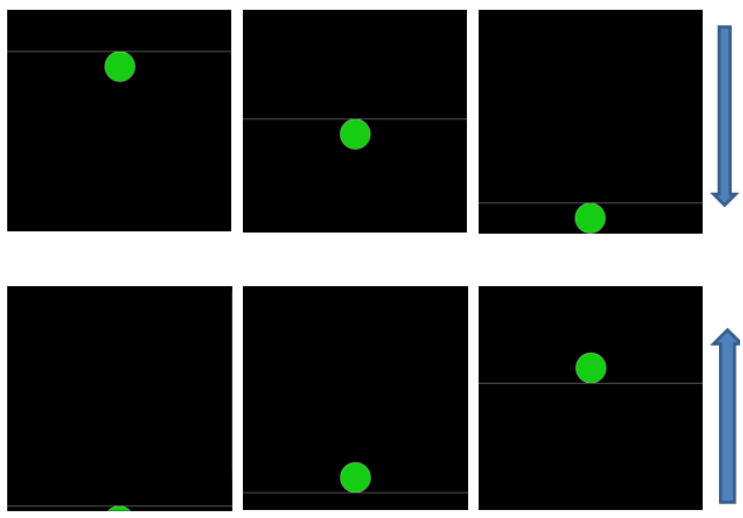
Analisis y Diseño (Ejercicios del 19 al 22)

Olarte Tatiana Magdalena

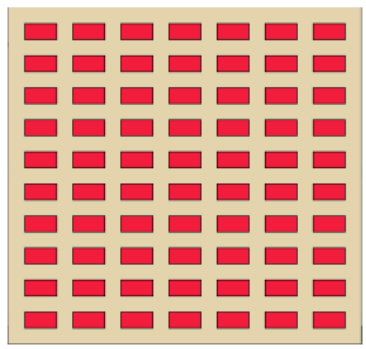
TUV000467

**Sección Análisis – Diseño y Codificación de algoritmos – Aplicación de estructuras de control**

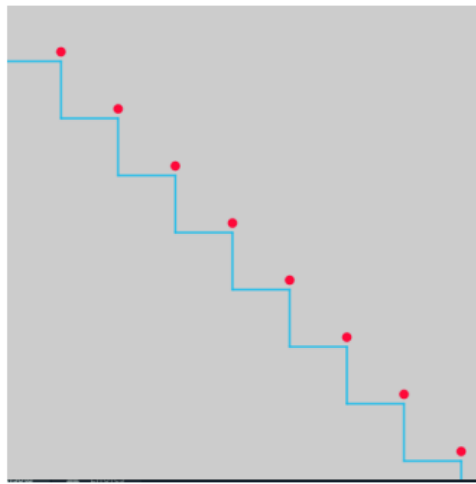
* Ejercicio 19: Declare las variables necesarias para dibujar una línea que se dibuja desde las coordenadas iniciales del lienzo y se extiende por todo el ancho. Sobre el punto medio de la línea y a una distancia de 40px (en sentido vertical desde la línea) dibuje una elipse que tenga como ancho 80px y de alto 80px. Dentro de la función draw(), actualice las variables necesarias para que la línea desde su inicio se mueva en dirección hacia abajo arrastrando la elipse. Mantenga en cero el valor para background(). Cuando la línea supere la posición de la altura del lienzo, debe invertir su sentido, es decir dirigirse hacia arriba arrastrando la elipse. Cuando la línea alcance nuevamente el valor 0 para su posición en y, el desplazamiento debe ser hacia abajo y así sucesivamente. El lienzo debería verse como en las siguientes figuras.



* Ejercicio 20: Dibuje en toda la extensión del lienzo de (440, 420) rectángulos de idénticas medidas (40 ancho y 20 de alto) y que mantengan una distancia de 20 pixeles entre ellos tanto horizontal como verticalmente. Utilice la estructura de control repetitiva for. El lienzo debería verse así:

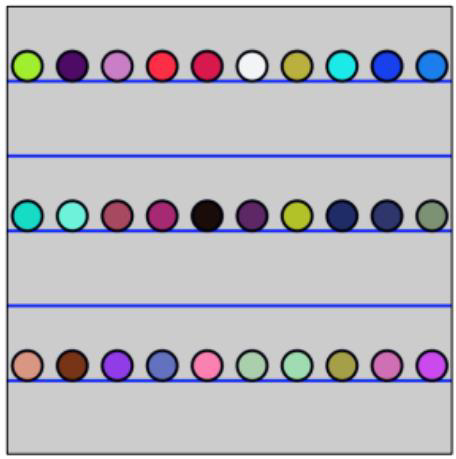


* Ejercicio 21: Utilizando la estructura de control repetitiva while() dibuje la siguiente imagen utilizando líneas que forman escalones y sobre cada borde de escalón se dibuje un punto de color rojo.



El tamaño del lienzo es size(500,500). La estructura while() se ejecuta dentro de la función setup(). La condición es que solo se dibuje dentro del lienzo. Utilice variables que puedan ayudar a la construcción del dibujo, por ej: x, y, anchoEscalon, altoEscalon, etc.

* Ejercicio 22: Utilizando la estructura de control repetitiva do-while. Replique la siguiente imagen:



La imagen debe ser construida desde la función setup(). Defina el tamaño del lienzo en size(600,600), verticalmente se divide el lienzo en franjas de igual medida, se deben dibujar los círculos sobre cada línea de por medio es decir en la línea 1 se dibujan círculos con distanciamiento, en la línea 2 no se dibuja y así sucesivamente. Las líneas tienen un color fijo, los círculos asumen colores aleatorios.

**DESARROLLO:**

* **Ejercicio 19:**

**Definición del Problema**: Se debe crear una línea horizontal que abarque todo el ancho del lienzo. Sobre el punto medio de la línea y a mitad del lienzo dibujar una elipse, donde ambas deberán moverse en dirección de abajo hacia arriba.

**Análisis:**

* Datos de Entrada: El ancho (80px) y alto (80px) de la elipse, el valor para background() y la distancia entre la linea y el centro de la elipse (40 px) en sentido vertical.
* Datos de Salida: El dibujo de la línea con la elipse moviéndose de abajo hacia arriba en el lienzo.
* Proceso:

¿Quién debe realizar el proceso?: El lienzo

¿Cuál es el proceso que realiza…? Primero se debe inicializar las variables que tenemos e ir dibujando la línea con la elipse en su debida posición, para después con la estructura de control selectiva IF ir desplazando los objetos de acuerdo con el enunciado.

**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  posY: Real // Almacena la posición en Y de la Elipse y la Línea  posX: Real // Almacena la posición en X de la Elipse y la Línea  anchoElipse: Real // Almacena el ancho que tendrá la Elipse  altoElipse: Real // Almacena el alto que tendrá la Elipse  velocidad: Real // Almacena la velocidad de desplazamiento de ambos objetos  distRadio: Real // Almacena la distancia desde la Línea hasta el centro de la elipse en sentido vertical o el Radio de la Elipse |
| **NOMBRE ALGORITMO**: dibujar\_linea\_y\_elipse    **PROCESO DEL ALGORITMO**   1. posX ← 0 2. posY ← 0 3. velocidad ← 1 4. distRadio ← 40 5. anchoElipse ← 80 6. altoElipse ← 80 7. Dibujar una linea en (posX,posY) con tamaño del ancho del lienzo 8. Dibujar una elipse en (posX + el ancho del lienzo/2, posY + distRadio) con dimensiones en ancho y alto 9. posY ← posY + velocidad 10. **si** (posY >= alto del Lienzo) **entonces** 11. velocidad (decremento 1) 12. distRadio ← distRadio – 40 13. **fin\_si** 14. **si** (posY <= 0) **entonces** 15. velocidad (incremento 1) 16. distRadio ← distRadio + 40 17. **fin\_si** |

* **Ejercicio 20:**

**Definición del Problema**: Se debe dibujar en todo el lienzo rectángulos que tengan la misma medida, así como también una distancia de 20 pixeles tanto vertical como horizontal.

**Análisis:**

* Datos de Entrada: El tamaño del lienzo (440, 420), las medidas de cada rectángulo (40 de ancho y 20 de alto) y la distancia ente cada uno (20 píxeles tanto horizontal como vertical).
* Datos de Salida: Todos los rectángulos que se pueden dibujar en el lienzo.
* Proceso:

¿Quién debe realizar el proceso?: El lienzo

¿Cuál es el proceso que realiza…? Se realiza con la estructura de control repetitiva **for** y con una estructura de control selectiva que es el **if,** para poder dibujar todos los rectángulos que se puedan en el lienzo.

**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  altoRect: Real // almacena el alto de los rectángulos  anchoRect: Real // almacena el ancho de los rectángulos  distanciaEnX: Real // almacena la distancia entre cada rectángulo en x  distanciaEnY: Real //almacena la distancia entre cada rectángulo en y  altoLienzo: Real // almacena la altura del lienzo  anchoLienzo: Real // almacena la anchura del lienzo  contRect: Real // almacena los rectángulos que se va haciendo  totalRect: Real // almacena el total de rectángulos que hay |
| **NOMBRE ALGORITMO**: dibujar\_rectangulos    **PROCESO DEL ALGORITMO**   1. anchoLienzo ← 440 2. altoLienzo ← 420 3. anchoRect ← 40 4. altoRect ← 20 5. distanciaEnX ← 20 6. distanciaEnY ← 20 7. totalRect ← 70 8. **para** contRect **←** 0 **hasta** totalRect **con paso** (incremento 1) **hacer** 9. Dibujar un rectangulo en (distanciaEnX,distanciaEnY) con dimensiones en ancho y alto 10. distanciaEnY ← distanciaEnY + altoRect \* 2 11. **si** (distanciaEnY > altoLienzo) **entonces** 12. distanciaEnY ← 20 13. distanciaEnX ← distanciaEnX + anchoRect \* 1.5 14. **fin\_si** 15. **fin\_para** |

* **Ejercicio 21:**

**Definición del Problema**: Se debe dibujar en el lienzo escalones formados por líneas y sobre cada borde un punto de color rojo.

**Análisis:**

* Datos de Entrada: El tamaño del lienzo (500,500), usar la estructura de control repetitiva WHILE dentro de la función setup() y que solo se dibuje dentro del lienzo.
* Datos de Salida: Los escalones con los puntos rojos sobre cada borde dibujados en el lienzo.
* Proceso:

¿Quién debe realizar el proceso?: El lienzo

¿Cuál es el proceso que realiza…? Se debe crear el primer escalón con el punto rojo en el borde de esta y asi sucesivamente, cambiando las posiciones tanto en los escalones hechos por líneas como en los puntos.

**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  posLinea: Coordenadas cartesianas en 2D // Almacena las posiciones X e Y de las lineas  posPunto: Coordenadas cartesianas en 2D // Almacena las posiciones X e Y de los puntos  anchoEscalon: Real // Almacena el ancho que tendrán los escalones  altoEscalon: Real // Almacena el alto que tendrán los escalones  distLinea: Real // Almacena la distancia que hay entre líneas  altoLienzo: Real // Almacena la altura del lienzo  anchoLienzo: Real // Almacena la anchura del lienzo |
| **NOMBRE ALGORITMO**: dibujar\_escalones\_y\_puntos    **PROCESO DEL ALGORITMO**   1. anchoLienzo ← 500 2. altoLinezo ← 500 3. posLinea ← en X 0, en Y 60 4. posPunto ← en X anchoEcalon, en Y anchoEscalon - 8 5. anchoEscalon ← 60 6. altoEscalon ← 60 7. distLinea ← 60 8. **mientras** (posLinea en X < anchoLienzo) **hacer** 9. Dibujar una linea en (posLinea en X,posLinea en Y) con dimensiones en ancho 10. Dibujar una linea en (posLinea en X + distLinea, posLinea en Y) con dimensiones en alto 11. posLinea en X ← posLinea en X + distLinea 12. posLinea en Y ← posLinea en Y + distLinea 13. anchoEscalon ← anchoEscalon + distLinea 14. altoEscalon ← altoEscalon + distLinea 15. **si** (anchoEscalon == altoEscalon) **entonces** 16. Dibujar un punto en (posPunto en X, posPunto en Y) 17. posPunto en X ← posPunto en X + distLinea 18. posPunto en Y ← posPunto en Y + distLinea 19. **fin\_si** 20. **fin\_mientras** |

* **Ejercicio 22:**

**Definición del Problema**: Se debe dibujar líneas que dividan en partes iguales al tamaño del lienzo en sentido vertical y sobre cada línea de por medio se debe dibujar círculos con distanciamiento parejo.

**Análisis:**

* Datos de Entrada: El tamaño del lienzo (600,600), usar la estructura repetitiva DO-WHILE desde la función setup(), se divide el lienzo en franjas de igual medida, se dibujan círculos sobre cada línea de por medio, las líneas tienen un color fijo y los círculos asumen colores aleatorios.
* Datos de Salida: La altura del lienzo dividido con líneas en partes iguales y sobre ellas los círculos con una separación de igual distancia. Todo esto dibujado en el mismo.
* Proceso:

¿Quién debe realizar el proceso?: El lienzo

¿Cuál es el proceso que realiza…? Se debe crear primero las líneas que dividen al lienzo verticalmente de forma pareja, para después sobre ellas dibujar los círculos con su respectiva distancia y color. Se debe usar la estructura de control do-while().

**Diseño:**

|  |
| --- |
| **ENTIDAD QUE RESUELVE EL PROBLEMA**: Lienzo |
| **VARIABLES**  posLinea: Coordenadas cartesianas en 2D // Almacena las posiciones X e Y de las lineas  posCirculo: Coordenadas cartesianas en 2D // Almacena las posiciones X e Y de los circulos  contCirculo: Real // Almacena la cantidad círculos que se van realizando  totalCirculo: Real //Almacena el total de circulos que hay que dibujar  tamañoCirculo: Real // Almacena el tamaño que tendrán los círculos  altoLienzo: Real // Almacena la altura del lienzo  anchoLienzo: Real // Almacena el ancho del lienzo |
| **NOMBRE ALGORITMO**: dibujar\_lineas\_y\_circulos    **PROCESO DEL ALGORITMO**   1. anchoLienzo ← 600 2. altoLinezo ← 600 3. posLinea ← en X 0, en Y altoLinezo/6 4. posCirculo ← en X 25, en Y (posLinea en Y) - 20 5. contCirculo ← 0 6. totalCirculo ← 30 7. tamañoCirculo ← 41 8. **hacer** 9. Dibujar una linea en (posLinea en X,posLinea en Y) con dimensiones en ancho 10. posLinea en Y ← posLinea en Y + altoLienzo/6 11. **si** (posCirculo en X < anchoLienzo) **entonces** 12. Dibujar un circulo en (posCirculo en X, posCirculo en Y) con dimensiones en ancho y alto 13. posCirculo en X ← posCirculo + 61 14. **fin\_si** 15. **si** (posCirculo en X > anchoLienzo) **entonces** 16. posCirculo en X ← 25 17. posCirculo en Y ← posCirculo en Y + altoLienzo/3 18. **fin\_si** 19. contCirculo (incremento 1) 20. **mientras** (contCirculo <= totalCirculo) 21. **fin\_hacer\_mientras** |

**CONCLUSIÓN:**

**FUENTES BIBLIOGRÁFICAS:**

* clase 03 09 04 24 Fudnamentos de Programación Orientada a objetos: <https://youtu.be/YHpW_HMP_PE?si=PmhigKipfKlWiWaD>
* 14 Estructuras de Control Iterativas: <https://virtual.unju.edu.ar/mod/resource/view.php?id=300666>
* 11 Bifurcaciones - Expresiones lógicas - Estructuras selectivas: <https://virtual.unju.edu.ar/mod/resource/view.php?id=299585>
* Clase 02 Aplicación Análisis y diseño operadores aritméticos y bifurcaciones: <https://youtu.be/bMJhygO43cA?si=UAbmVd5YzrE0Drhi>
* 14 Estructuras de Control Iterativas: <https://virtual.unju.edu.ar/mod/resource/view.php?id=300666>
* PROCESSING | Estructuras de control | Condicionales | Programación: <https://youtu.be/Nr3NdAxjqsE?si=dGYWzigsIKYv0ap5>
* Ejemplo analisis-diseño-codificación en Processing: <https://youtu.be/_YblzDgoAus?si=vGTjzoQgeXeKzvMG>