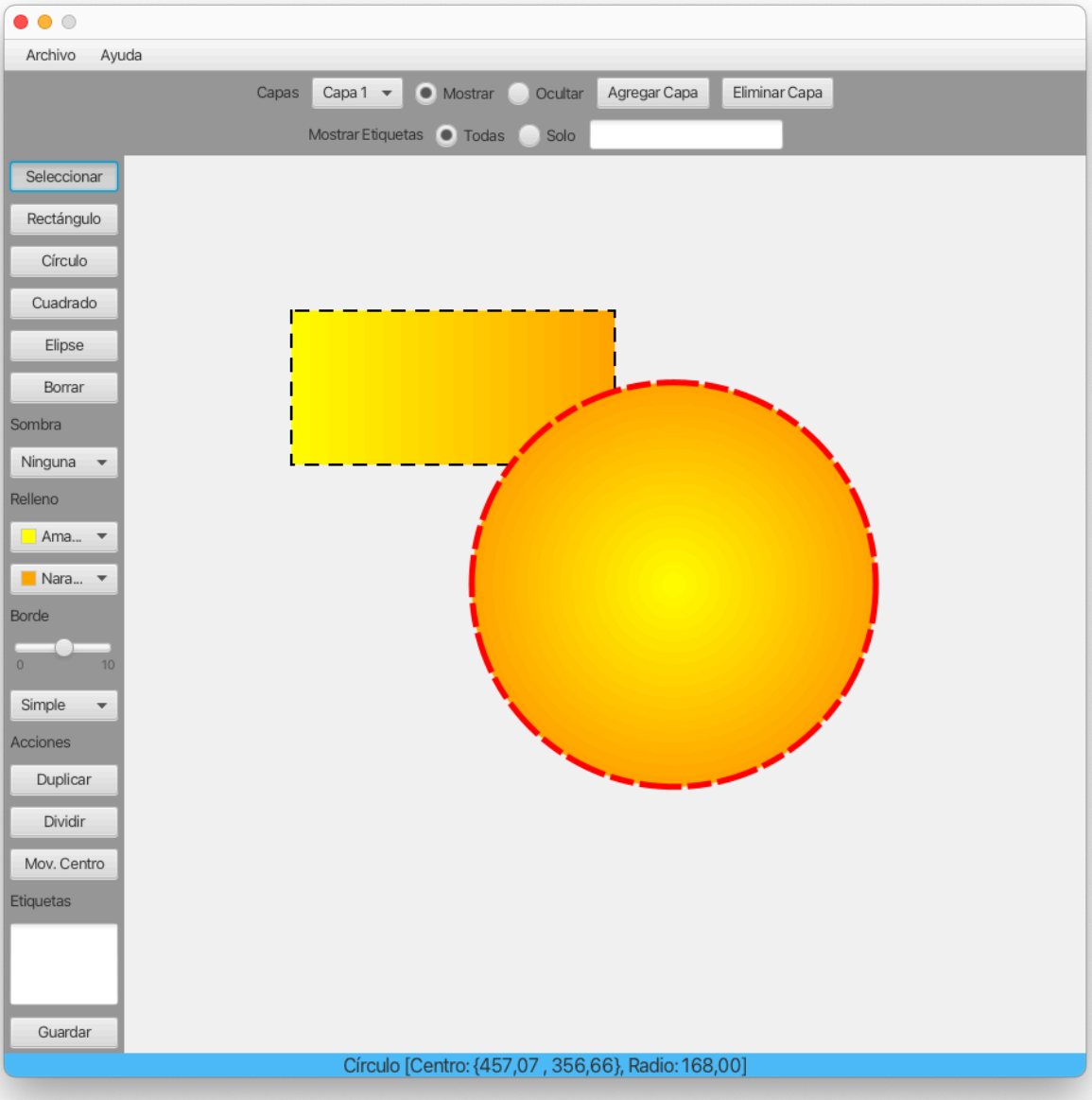
Trabajo Práctico Especial
Programación Orientada a Objetos
Julio 2024

Objetivo

El objetivo del presente trabajo es implementar una **aplicación de dibujo de figuras**, desarrollado en lenguaje Java, con interfaz visual en JavaFX, aplicando los conceptos sobre diseño orientado a objetos adquiridos en la materia.



Descripción funcional

Se cuenta con una implementación inicial de la aplicación donde se pueden dibujar rectángulos, cuadrados, elipses y círculos que pueden moverse en el espacio 2D.

Para dibujar un rectángulo se debe presionar el botón Rectángulo de la barra lateral izquierda, hacer click en el área de dibujo para definir el punto superior izquierdo del mismo, mantener presionado el mouse y arrastrarlo hasta el punto inferior derecho. Una vez que se dejó de presionar el mouse el rectángulo aparecerá dibujado en la pantalla. Para dibujar un cuadrado se define primero el punto superior izquierdo y luego se arrastra el mouse hasta el ancho deseado. Para dibujar un círculo debe seleccionar el botón Círculo, hacer click en el área de dibujo para definir el centro del círculo, mantener presionado el mouse y arrastrarlo hasta un punto, donde el radio será la diferencia entre el punto final y el centro del círculo (en el eje x). Para dibujar una elipse se debe dibujar el rectángulo que encierra por completo a la elipse.

Para mover una figura se debe presionar el botón Seleccionar de la barra izquierda, hacer click en una figura y luego mantener presionado el mouse para mover la figura en la dirección del cursor. Nótese que cuando una figura es seleccionada, su borde cambia. Para mover una figura seleccionada no es necesario que el click inicial antes de mantener presionado esté dentro de la figura en cuestión, puede estar en cualquier parte. Es posible mover una figura de forma que termine quedando fuera del canvas y ese es el comportamiento esperado.

Para borrar una figura se debe presionar el botón Borrar teniendo una figura seleccionada.

En la parte inferior de la ventana se ofrece una línea de texto con información relevante. Si mueve el cursor puede verse las coordenadas del mismo en ese instante. Y si el cursor pasa por encima de una figura se indica información de la misma.

El color de relleno de la figura a dibujar (por defecto un amarillo) se puede elegir en el selector de color de la barra izquierda. Este componente no debe reflejar el color de la figura seleccionada, es sólo para elegir el color de la figura a dibujar. Una vez dibujada una figura ésta no puede cambiar su color.

Desarrollo del trabajo

En Campus ITBA, en la carpeta **Final Julio 2024** del Contenido del Curso se encuentra el enlace del proyecto Java con los fuentes de esta aplicación.

Como primer paso el grupo debe analizar todos los fuentes para comprender el diseño y las clases que lo componen y entender cada detalle de funcionamiento.

Como segundo paso el grupo debe realizar modificaciones a los fuentes provistos siguiendo los lineamientos dictados en clase. Adrede, en varias partes del código provisto existen ejemplos de malas prácticas que violan los principios del paradigma POO y que deberán ser corregidos por parte del grupo.

Como tercer paso, el grupo deberá agregar nuevas funcionalidades a la implementación provista.

En todos los casos, se espera que la implementación de la nueva funcionalidad **demuestre las ventajas del paradigma**. Y también se espera una **correcta separación entre el *front-end* y el *back-end***: el *back-end* de esta implementación

debería poder ser utilizado también por otros *front-end* además de JavaFX (*mobile*, web, etc.)

En caso de rendir en la primera fecha de final y formando un grupo de tres o menos alumnos, deberán implementar las tres primeras funcionalidades.

En caso de rendir en la primera fecha de final y formando un grupo de cuatro alumnos, deberán implementar las cuatro primeras funcionalidades.

En caso de rendir en la segunda fecha de final deberán formar grupos de tres o menos alumnos e implementar TODAS las funcionalidades. No se aceptarán grupos de cuatro integrantes para la segunda fecha.

1. Sombra, Relleno y Borde

1.1 Sombra

Modificar el comportamiento actual para que se puedan elegir entre **cuatro tipos de sombras** de una figura antes de dibujarla y para poder cambiar la sombra de cualquier figura ya dibujada. Se debe poder seguir dibujando figuras sin sombra y se debe poder quitar la sombra de una figura ya dibujada.

El efecto de **sombra** consiste en **dibujar primero la sombra y luego la figura encima**.

Los cuatro tipos de sombras son:

- **Simple**: Dibujar la sombra consiste en dibujar la figura desplazada en un **offset** positivo utilizando el color gris.
- **Coloreada:** Dibujar la sombra consiste en dibujar la figura desplazada en un **offset** positivo utilizando una versión oscurecida del color de relleno de la figura
- **Simple Inversa:** Dibujar la sombra consiste en dibujar la figura desplazada en un **offset** negativo utilizando el color gris.
- **Coloreada Inversa:** Dibujar la sombra consiste en dibujar la figura desplazada en un **offset** negativo utilizando una versión oscurecida del color de relleno de la figura

Por ejemplo, para conseguir la sombra **Simple** de un círculo en la implementación del método `redrawCanvas` se puede hacer lo siguiente:

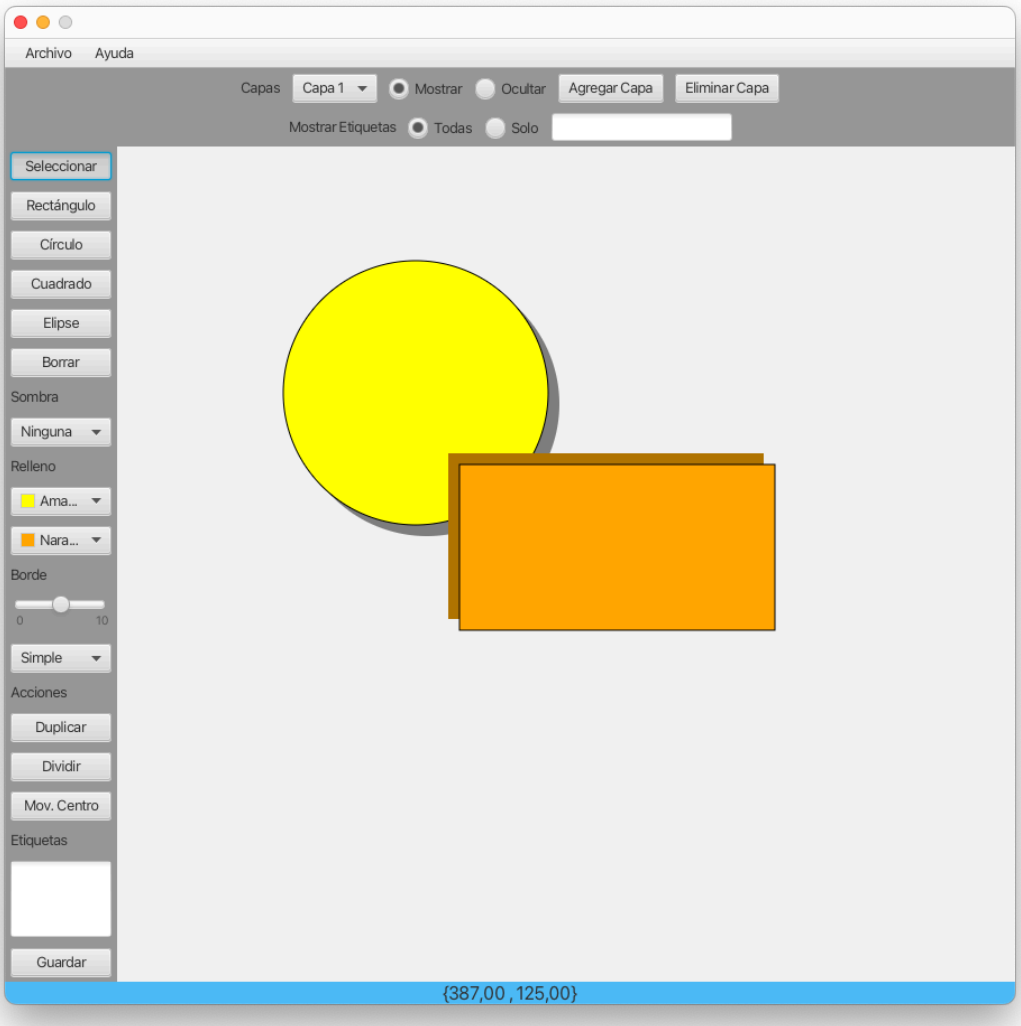
```
gc.setFill(Color.GRAY);
gc.fillOval(circle.getCenterPoint().getX() - circle.getRadius() + 10.0,
      circle.getCenterPoint().getY() - circle.getRadius() + 10.0, diameter,
diameter);
```

(donde 10.0 es un valor fijo de offset positivo en X y en Y y el color gris es un color fijo) y para conseguir la sombra **Coloreada Inversa** de un rectángulo en la implementación del método `redrawCanvas` se puede hacer lo siguiente:

```
gc.setFill(figureColorMap.get(rectangle).darker());
gc.fillRect(rectangle.getTopLeft().getX() - 10.0,
      rectangle.getTopLeft().getY() - 10.0,
      Math.abs(rectangle.getTopLeft().getX() -
rectangle.getBottomRight().getX()),
      Math.abs(rectangle.getTopLeft().getY() -
```

```
rectangle.getBottomRight().getY()));
```

(donde 10.0 es un valor fijo de offset negativo en X y en Y y se utiliza el método `darker` para oscurecer el color de relleno). El efecto logrado es el siguiente:



1.2 Relleno

Modificar el comportamiento actual para que se puedan elegir **dos colores para el relleno** de una figura antes de dibujarla y para poder cambiar uno o los dos colores de relleno de cualquier figura ya dibujada.

Para lograr un relleno que fusione los dos colores elegidos utilizar una instancia de `LinearGradient` (para los rectángulos) y `RadialGradient` (para las elipses). Esto creará un relleno que resulta en la transición entre el primer color de relleno y el segundo color de relleno.

Por ejemplo, para conseguir el relleno de un círculo en la implementación del método `redrawCanvas` se puede hacer lo siguiente:

```
RadialGradient radialGradient = new RadialGradient(0, 0, 0.5, 0.5, 0.5, true,
      CycleMethod.NO_CYCLE,
```

```
      new Stop(0, firstFillColor),
      new Stop(1, secondFillColor));
gc.setFill(radialGradient);
```

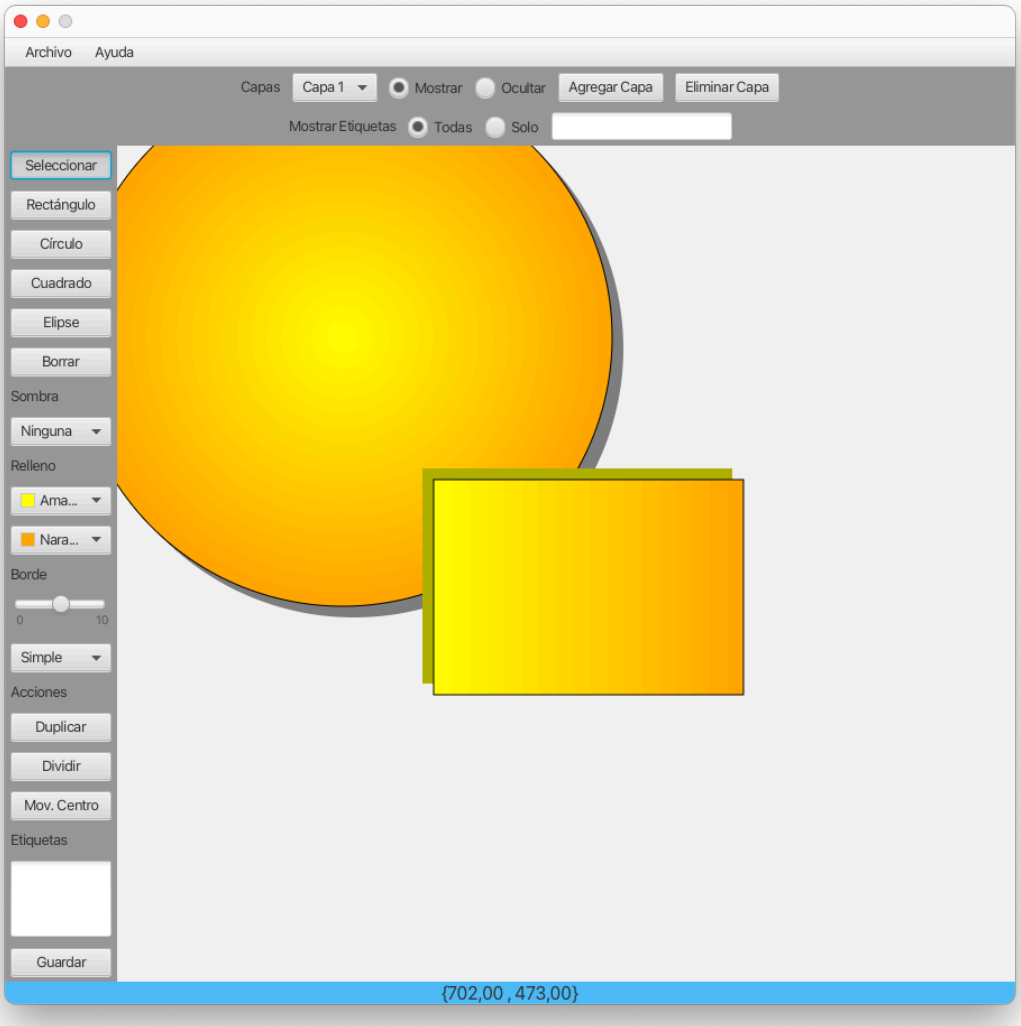
para que la siguiente invocación a `gc.fillOval` utilice al gradiente como color de relleno.

Para conseguir el relleno de un rectángulo en la implementación del método `redrawCanvas` se puede hacer lo siguiente:

```
LinearGradient linearGradient = new LinearGradient(0, 0, 1, 0, true,
      CycleMethod.NO_CYCLE,
      new Stop(0, firstFillColor),
      new Stop(1, secondFillColor));
gc.setFill(linearGradient);
```

para que la siguiente invocación a `gc.fillRect` utilice al gradiente como color de relleno.

En ambos los valores 0, 1, 0.5 son fijos para lograr el siguiente resultado (donde el primer color de relleno de la figura es un color amarillo y el segundo color de relleno es un color naranja):



Respecto a ***1.1 Sombra*** elegir el primer color de relleno como referencia para oscurecer el color de los tipos de sombra Coloreada y Coloreada Inversa.

1.3 Borde

Modificar el comportamiento actual para que se puedan elegir entre **tres tipos de bordes** además de definir el **ancho para los bordes** de una figura antes de dibujarla y para poder cambiar el tipo y ancho de los bordes de cualquier figura ya dibujada.

Respecto al ancho del borde utilizar el método **`GraphicsContext#setLineWidth`**.

Los tres tipos de bordes son:

- **Normal**: La línea continua de la implementación original
- **Punteado simple:** Un patrón uniforme de "-" separados por espacios, es decir, todos del mismo ancho
- **Punteado complejo:** Un patrón de "--" y "-" separados por espacios, es decir, uno de ancho simple seguido de uno de doble ancho, seguido de uno de ancho simple, etc.

Para lograr los tipos de borde punteado utilizar el método **`GraphicsContext#setLineDashes`** que espera una serie de números que definen la longitud de los guiones y los espacios en el patrón, en ese orden específico.

Por ejemplo, para conseguir el tipo de borde Punteado simple en la implementación del método `redrawCanvas` se puede hacer lo siguiente:

```
gc.setLineDashes(10);
```

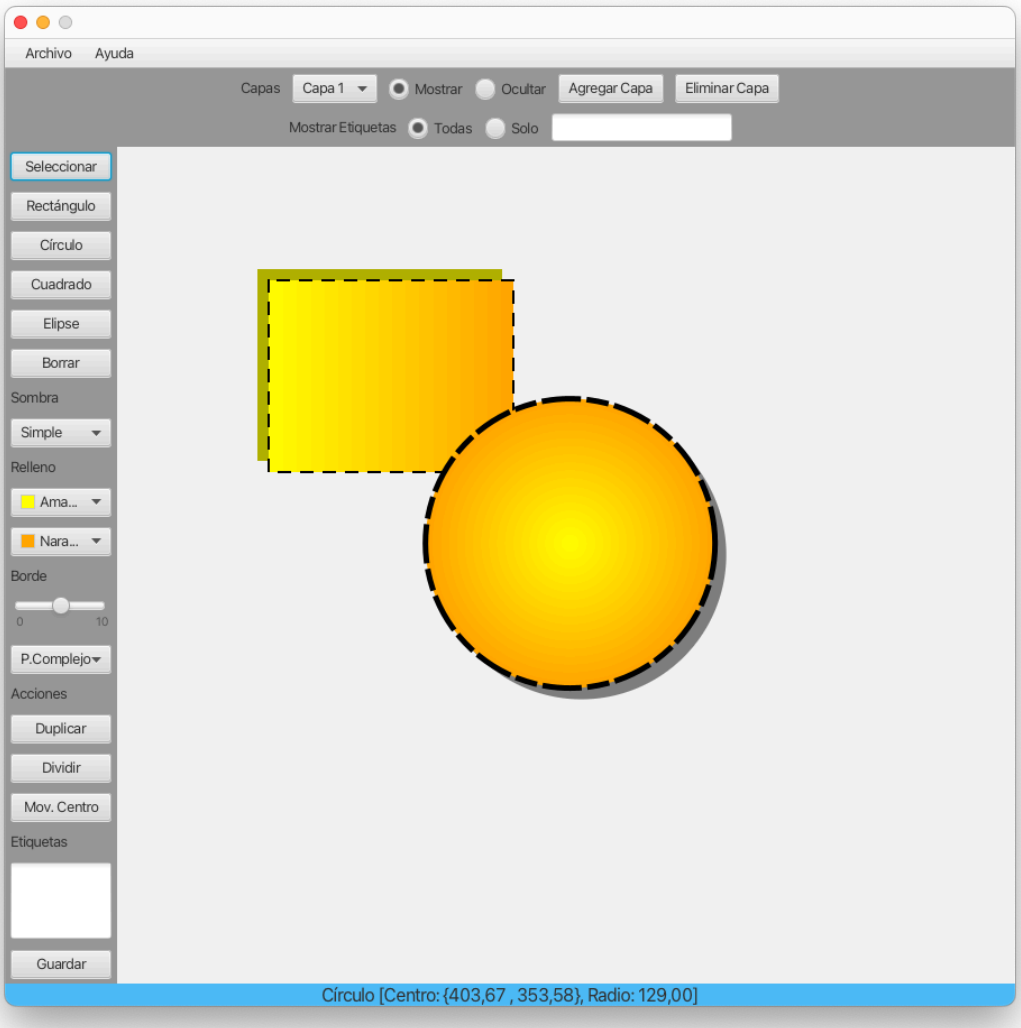
para que la siguiente invocación a `gc.strokeRect` o `gc.strokeOval` utilice ese tipo de borde, donde 10 es el ancho del espacio.

Para conseguir el tipo de borde Punteado complejo en la implementación del método `redrawCanvas` se puede hacer lo siguiente:

```
gc.setLineDashes(30, 10, 15, 10);
```

para que la siguiente invocación a `gc.strokeRect` o `gc.strokeOval` utilice ese tipo de borde, donde 10 es el ancho del espacio, 30 es el ancho de patrón de doble ancho y 15 es el ancho del patrón de ancho simple.

En ambos casos los valores 10, 15 y 30 son fijos para lograr el siguiente resultado (donde el rectángulo tiene un borde de ancho 2 y de tipo Punteado simple 2 y el círculo tiene un borde de ancho 5 y de tipo Punteado complejo):



1.4 Consideraciones Generales

Agregar a la barra lateral izquierda tres nuevas secciones **Sombra**, **Relleno** y **Borde**. En la sección de sombra se debe poder elegir el tipo de sombra de la figura con un `ChoiceBox` antes de dibujarla (una opción más del `ChoiceBox` debería ser Ninguna). En la sección de relleno se debe poder elegir el primer color de relleno de la figura y el segundo color de relleno de la figura antes de dibujarla, ambos con un `ColorPicker`. En la sección de borde se debe poder elegir el grosor del borde de la figura con un `Slider` y el tipo del borde de la figura con un `ChoiceBox` antes de dibujarla.

En cuanto al ***front-end***, recomendamos consultar la documentación de las clases:

- `javafx.scene.control.Label`
- `javafx.scene.paint.Color`
- `javafx.scene.control.ColorPicker`
- `javafx.scene.control.Slider`
- `javafx.scene.control.ChoiceBox`

Estos controles deben poder modificar la sombra, relleno y borde de figuras ya creadas. En caso de seleccionar una figura en pantalla, los controles deben reflejar las propiedades de sombra, relleno y borde de la figura seleccionada. Al seleccionar una figura si luego se cambia el primer color de relleno en la barra izquierda, se modifica el primer color de relleno de la figura

seleccionada. Y en cualquier momento en que se seleccione una figura, el color del borde cambiará a rojo.

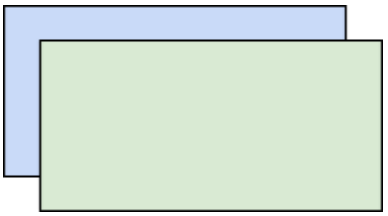
Deberán definir dos colores de relleno por defecto. Los componentes de selección de color de la barra izquierda deberán reflejar estos colores al iniciar la aplicación. Cada vez que se crea una nueva figura esta deberá tomar los colores de relleno que están a la vista en los `ColorPicker` de la barra izquierda. En caso de dibujar una figura pequeña es posible que, al definirle un valor muy alto de grosor del borde, el relleno de la figura no se aprecie y este es el comportamiento esperado. Y en caso de elegir el mismo color para los dos colores de relleno, no se apreciará ningún efecto y este es el comportamiento esperado.

2. Duplicar, Dividir y Mover al Centro

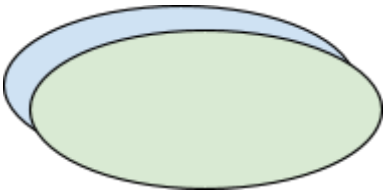
2.1 Duplicar

Agregar un botón **Duplicar** en la barra lateral izquierda para duplicar una figura seleccionada.

A continuación un ejemplo de un rectángulo (celeste) y ese rectángulo duplicado (verde), donde las coordenadas X e Y de la nueva figura difieren en un offset fijo a definir en la implementación.



Otro ejemplo, en este caso para una elipse (celeste) y esa elipse duplicada (verde):

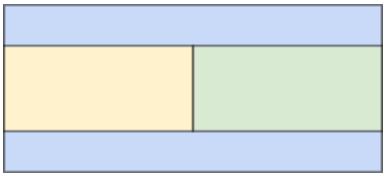


Si no hay figura seleccionada, presionar el botón no tiene efecto.

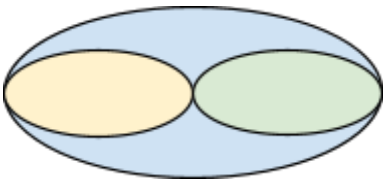
2.2 Dividir

Agregar un botón **Dividir** en la barra lateral izquierda para dividir una figura seleccionada. Dividir una figura eliminará la figura original y creará dos nuevas con la mitad de las dimensiones, respetando las coordenadas de la original.

A continuación un ejemplo de un rectángulo (celeste) que se divide en dos rectángulos (amarillo y verde). El rectángulo que fue dividido (celeste) no debe dibujarse.



Otro ejemplo, en este caso para una elipse (celeste) que fue divida en dos elipses (amarillo y verde). La elipse que fue dividida (celeste) no debe dibujarse.



Si no hay figura seleccionada, presionar el botón no tiene efecto.

Debido a los bordes y sus anchos es posible que luego de dividir una figura estas nuevas figuras se "solapen" y este es el comportamiento esperado.

2.3 Mover al centro

Agregar un botón **Al Centro** en la barra lateral izquierda para mover una figura seleccionada al centro del lienzo.

Si no hay figura seleccionada, presionar el botón no tiene efecto.

2.4 Consideraciones Generales

Respecto a la implementación de ***1. Sombra, Relleno y Borde*** al duplicar o dividir una figura las nuevas figuras que se generen sombra, rellenos y borde de la figura original (eso sí, aplicado a los nuevos puntos y dimensiones que tendrá la figura cambiada).

3. Capas: Crear y Mostrar

Agregar a la barra superior una nueva sección **Capas** y un nuevo `ChoiceBox` para elegir una capa. Se debe poder elegir una capa antes de dibujar una figura. Luego la figura recién dibujada pertenecerá a esa capa y no podrá cambiar de capa.

La aplicación inicia con tres capas "Capa 1", "Capa 2" y "Capa 3". Agregar un botón **Agregar Capa** para crear una nueva capa (si es la primera vez que se presiona en la aplicación creará la capa "Capa 4"). Al agregar una nueva capa ésta se elige automáticamente de forma que de dibujar una nueva figura esta pertenecerá a la capa recién creada. Agregar además dos `RadioButton` para indicar si la capa elegida (la que se muestra en el `ChoiceBox`) está visible u oculta. Por ejemplo, cuando se tilda el `RadioButton` que dice **Ocultar** habiendo elegido la "Capa 1" se deben ocultar las figuras de la "Capa 1". Y cuando se tilda el `RadioButton` que dice **Mostrar** habiendo elegido la "Capa 1" se deben mostrar las figuras de la "Capa 1". Por último agregar un botón **Eliminar Capa** que permite eliminar la capa elegida. Esta acción implica eliminar todas las figuras de la capa y la acción no se puede deshacer. Si se presiona Eliminar Capa habiendo elegido alguna de las tres capas iniciales la acción no tiene efecto (esas tres capas iniciales no se pueden eliminar). Luego de eliminar capas se pueden obtener saltos en la numeración de las capas de la aplicación y es válido. Por ejemplo si se presionó el botón Agregar Capa dos veces y se elige la "Capa 4" y se presiona Eliminar Capa las capas serán "Capa 1", "Capa 2", "Capa 3" y "Capa 5".

Las tres capas iniciales arrancan por defecto como visibles (opción Mostrar). Cada vez que se agrega una capa ésta también arranca visible.

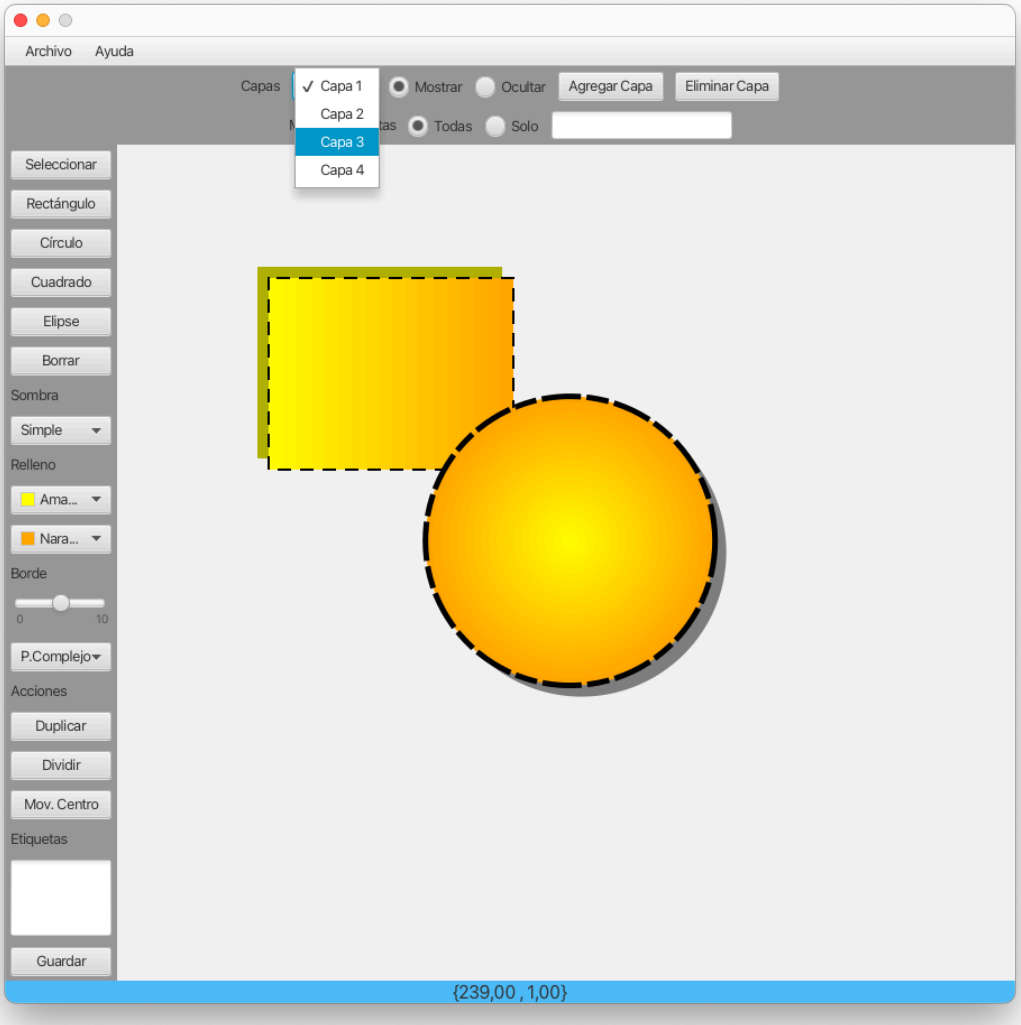
Además de la pertenencia, las capas le agregan un comportamiento de profundidad a las figuras. Todas las figuras de una capa superior están por encima de las figuras de una capa inferior

(es decir, todas las figuras de la “Capa 3” se deben dibujar por encima de las figuras de la “Capa 2”).

En cuanto al ***front-end***, recomendamos consultar la documentación de las clases y métodos:

- `javafx.scene.control.Label`
- `javafx.scene.control.ChoiceBox`
- `javafx.scene.control.RadioButton`
  - `javafx.scene.control.ToggleGroup`
- `javafx.scene.control.Button`

A continuación un ejemplo de como quedarían los componentes necesarios de esta funcionalidad en la barra superior:



4. Etiquetas: Mostrar y Ocultar

Agregar a la barra lateral izquierda una nueva sección **Etiquetas** y un nuevo `TextArea` para indicar las etiquetas o *tags* que le corresponden a una figura. Una vez seleccionada una figura se debe reflejar en el cuadro de texto la/s etiqueta/s que tiene esa figura. Una figura se dibuja inicialmente sin etiquetas. Para agregarle una etiqueta a una figura basta con seleccionarla, escribir la/s etiqueta/s en el cuadro de texto y presionar el botón de **Guardar**. Para guardar múltiples etiquetas deberá procesar el contenido del cuadro de texto y separarlo en los espacios, donde cada palabra es una etiqueta. Si se ingresa el texto “rect simple” o “rect**n**simple” y se presiona guardar

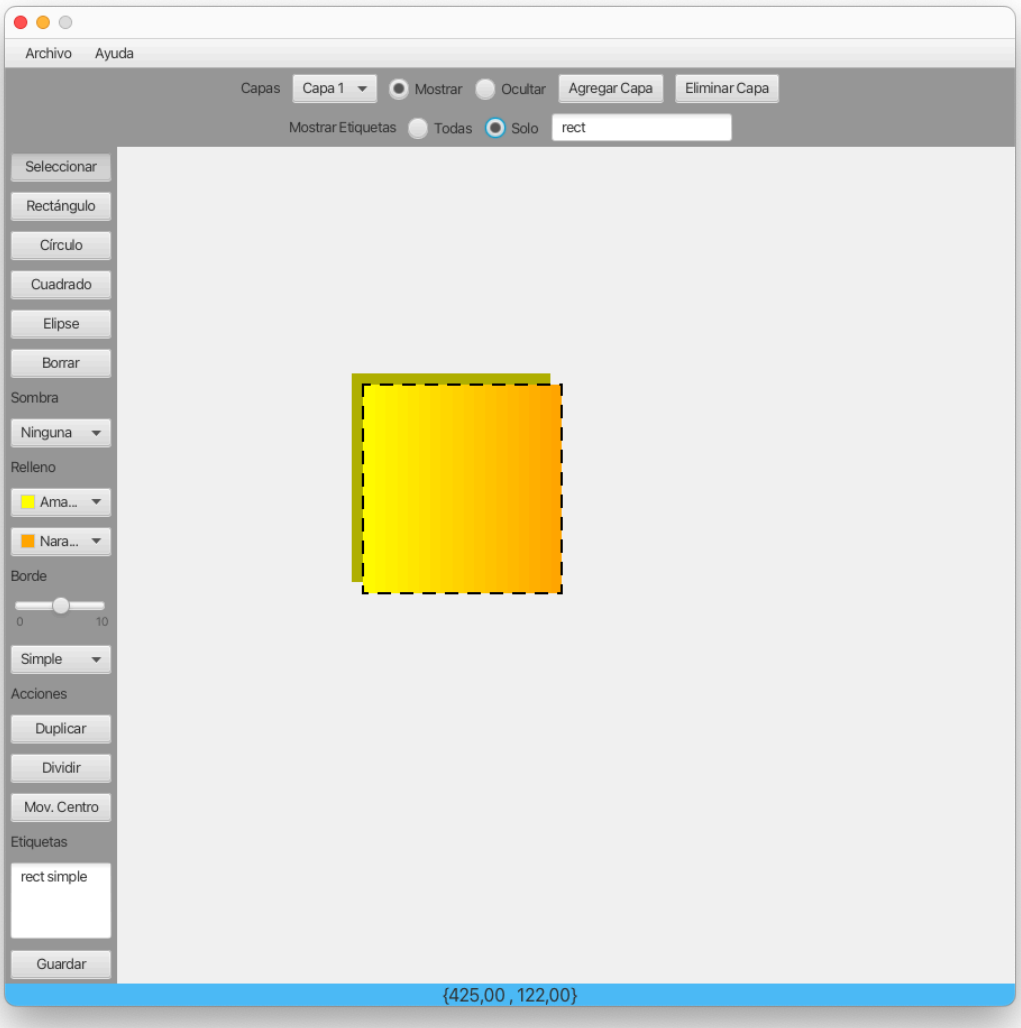
entonces la figura seleccionada ahora cuenta con dos etiquetas: “rect” y “simple”. Si una figura seleccionada ya contaba con etiquetas y se presiona el botón Guardar entonces se reemplazan las etiquetas existentes por las presentes en el cuadro de texto.

Agregar también a la barra superior una nueva sección **Mostrar Etiquetas** y dos nuevos `RadioButton` para poder filtrar sólo las figuras que tengan cierta etiqueta ingresada en el nuevo `TextField` o para mostrar todas. Cuando se hace click en la opción Todas no se debe ocultar ninguna figura. Cuando se hace click en la opción Sólo se deben ocultar todas las figuras que no tengan la etiqueta ingresada en el cuadro de texto. Sólo interesa la primera palabra ingresada en el cuadro de texto, es decir, se ignora lo que se ingresa después del espacio pues la funcionalidad es mostrar y ocultar en función de una única etiqueta.

El filtro de etiquetas debe ser compatible con el de la funcionalidad ***3. Capas: Crear y Mostrar***. Si por ejemplo se selecciona la opción “Solo” habiendo ingresado el texto “rect” y sólo se indicó mostrar las figuras de las capas “Capa 1” y “Capa 2” entonces se deben ocultar todas las figuras de las demás capas y todas las figuras que no tengan la etiqueta “rect” pertenecientes a las capas “Capa 1” y “Capa 2”. Si se indicó ocultar las figuras de todas las capas y se selecciona la opción “Todas” de Mostrar Etiquetas entonces se deben ocultar todas las figuras.

En cuanto al ***front-end***, recomendamos consultar la documentación de las clases y métodos:

- `javafx.scene.control.Label`
- `javafx.scene.control.TextArea`
- `javafx.scene.control.Button`
- `javafx.scene.control.RadioButton`
    - `javafx.scene.control.ToggleGroup`
        - `selectedToggleProperty()`
- `javafx.scene.control.TextField`



5. JUnit 5: Tests de Unidad

Se deben implementar tests de unidad utilizando el *framework* de testeo JUnit 5 cubierto en clase. **Todos los métodos de las clases de *back-end* deben ser invocados al menos una vez en los test de unidad.** Y un test unitario puede testear varios métodos distintos.

No deben testear los métodos de las clases de *front-end*.

Grupos

Los alumnos deben informar la conformación del grupo y la fecha de final elegida en el Debate **"TPE Final: Conformación Grupos"** de Campus ITBA **al menos un día antes de la fecha de entrega correspondiente**.

Cada grupo deberá realizar la entrega mediante la actividad correspondiente en Campus ITBA:

- **En caso de rendir en la primera fecha de final (11/07/2024) deberán entregar antes del 08/07/2024 23:59 ART**
- **En caso de rendir en la segunda fecha de final (18/07/2024) deberán entregar antes del 15/07/2024 23:59 ART**

El grupo completo deberá presentarse en un día y hora previamente acordada (se asignarán turnos para todos los grupos que rindan el final ese día). En ese encuentro se comunicará la nota del final y los integrantes del grupo rendirán un coloquio, el cual podrá modificar la nota individual de los integrantes del grupo.

Para que el trabajo sea aceptado todos los integrantes del grupo deben estar inscriptos en fecha de final que acordaron en el armado del grupo.

En caso de inscribirse y no entregar en fecha, se calificará como ausente. Si algún miembro del equipo no se presenta en el horario previamente acordado el alumno será calificado como ausente, y deberá conformar otro grupo en otra fecha de final, no afectando la evaluación de los alumnos que se presenten.

***Importante*: TODOS los integrantes del grupo DEBEN inscribirse por su cuenta en la fecha de final elegida en el SGA. Esto implica que todos cuentan con todas las correlativas necesarias para ello.**

Uso de Git

Es obligatorio el uso de un repositorio Git para la resolución de este final. Deberán crear un repositorio en GitHub donde todos los integrantes del grupo colaboren con las modificaciones del código provisto. **No se aceptarán entregas que utilicen un repositorio *git* con un único *commit* que consista en la totalidad del código a entregar**.

Los distintos *commits* deben permitir ver la evolución del trabajo, tanto grupal como **individual**

Muy importante: los repositorios creados deben ser privados.

Material a entregar

Se deberá entregar en un archivo comprimido el proyecto con las modificaciones y además un breve informe que contenga:

- Cambios hechos en la implementación provista por la Cátedra, justificando si fue para corregir un error de funcionamiento, mejorar la eficiencia o el estilo.
- Enumeración y breve descripción de las funcionalidades agregadas: no es necesario mencionar en qué consiste la funcionalidad sino cómo las implementaron
- Problemas encontrados durante el desarrollo

  El informe no debe superar las tres páginas (excluyendo diagramas UML).

  Revisar que en el archivo comprimido también se encuentra el directorio oculto .git/ donde se detallan todas las modificaciones realizadas.

Criterios de evaluación y calificación

No se aceptará el uso de bibliotecas de terceros, a excepción de la biblioteca estándar de Java. El código debe ser íntegramente de autoría propia. El uso de bibliotecas no autorizadas implicará la desaprobación.

No se aceptarán soluciones que utilicen clases de los paquetes `javafx.scene.shape`, `java.awt.geom` y similares. Es parte de la evaluación de este trabajo el diseño de las clases de figuras que sean necesarias.

El programa no debe abortar por ningún motivo y ante cualquier error se debe mostrar un mensaje adecuado.

Para la evaluación y calificación del trabajo especial se considerarán:

- el correcto funcionamiento del programa
- la modularización realizada
- el correcto diseño de las clases
- la separación del *front-end* y el *back-end*
- la claridad del código
- el cumplimiento de las reglas de estilo de programación dadas en clase

Se evaluará además el directorio .git/ entregado para verificar un desarrollo progresivo y con participación de todos los integrantes del grupo.

Enlaces Útiles

- Part II: Using JavaFX UI Controls (Release 8)
- JavaFX Ensemble de JavaFX Samples

Dudas sobre el TPE

Si bien el enunciado contempla la funcionalidad completa a desarrollar es normal que surjan dudas acerca de cómo interpretar ciertos casos. O que una consigna genere más de una posible solución, por lo que es importante que analicen bien el enunciado, y ante cualquier duda pregunten. Sólo se contestarán dudas sobre el enunciado. Las mismas deben volcarse en Github: https://github.com/POO-ITBA/2024_01/issues

En caso de realizar alguna aclaración o consideración sobre el enunciado, la misma deberá ser tenida en cuenta por todos los grupos, no solo para el grupo que haya hecho la pregunta.