Trabajo Práctico Especial
72.33 - Programación Orientada a Objetos

Fecha de entrega: 11/72024

Integrantes:

Keoni Dubovitsky, 62815

Lucas Di Candia, 63212

Franco Ferrari, 63094

1. Cambios en la implementación original y funcionalidades

Herencia de figuras

Se cambió el back de manera tal que la clase *Square* hereda a *Rectangle,* ya que un cuadrado es un caso particular de un rectángulo con todos sus lados iguales. De misma manera con la clase *Circle* y *Ellipse.*

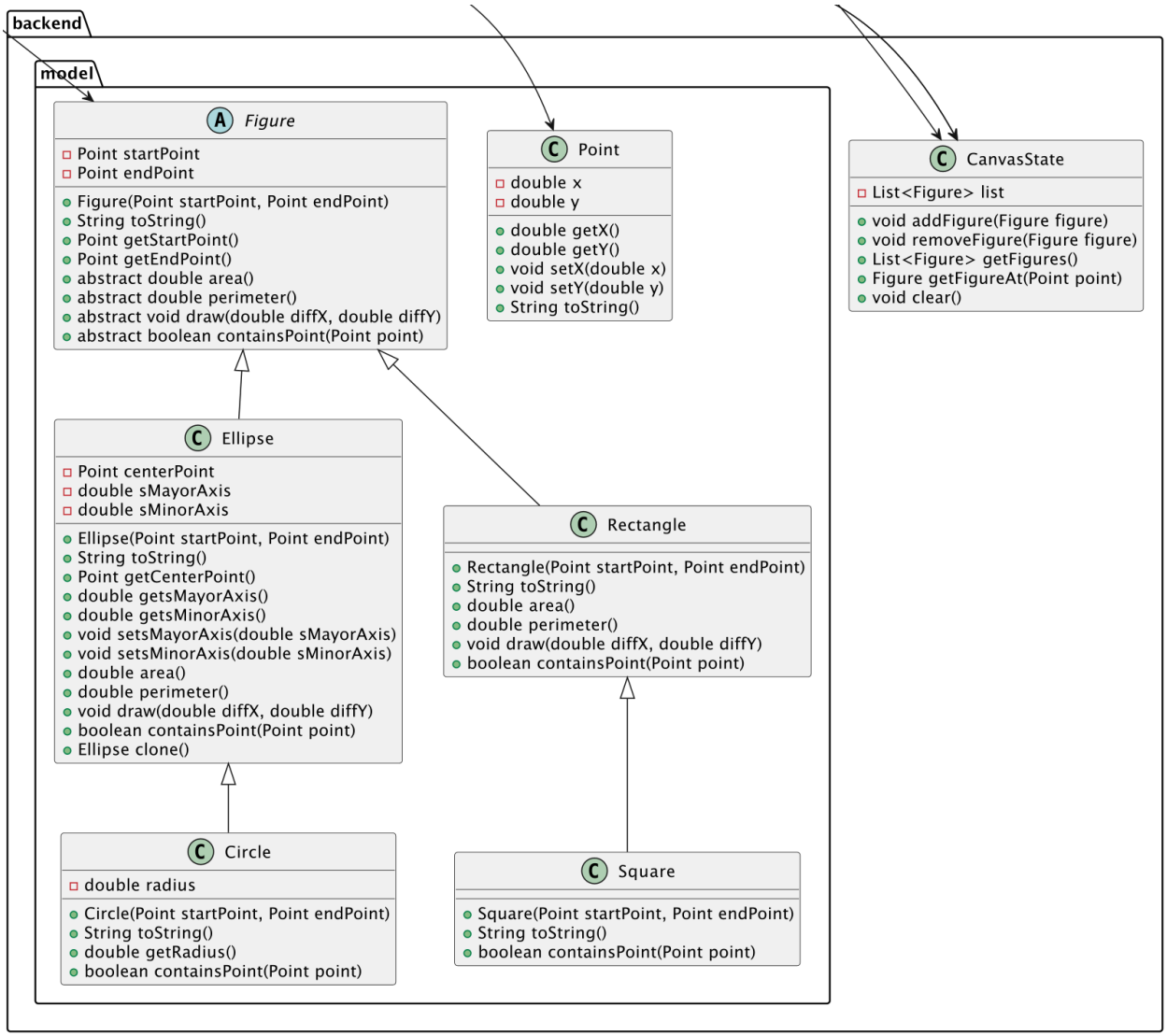

Diagrama UML mostrando los contenidos de backend

Mejora de eficiencia

Se optimizó el manejo de eventos y el redibujado del lienzo, minimizando las operaciones redundantes y mejorando la fluidez del rendimiento de la aplicación. Esto implicó la refactorización de los métodos de actualización en *CanvasState.*

2. Cambios en la implementación original y funcionalidades

Implementación de la clase *CanvasPane*

Dado que la clase original PaintPane contenía numerosos ejemplos de malas prácticas, decidimos reemplazarla con CanvasPane. Esta nueva clase se encarga del dibujo de las figuras y la gestión de los diferentes modos activados por los botones de la interfaz.

Dependiendo del modo seleccionado, CanvasPane realiza las acciones, de dibujar y eliminar figuras, ademas de un comparador para sortear por layers.

Implementación de la clase *FigureRenderer*

Esta clase es abstracta y es padre de todas las figuras dibujables, lo que nos permite organizar de mejor manera la lógica de dibujo y aprovechar herramientas del paradigma como la herencia que facilita la reutilización de código.

Implementacion de los botones

Se armaron 5 tipos distintos de botones, que implementan las distintas funcionalidades, los botones son, *ChoiceBox,* sirve para elegir el tipo de linea, de sombra y de capa. Despues tambien esta *ColorPicker*, este sirve para elegir el tipo de color con el que se quiere llenar la figura. Tambien esta *RadioButton*, el cual sirve para ocultar o mostrar una capa en especifico. Aparte esta el *Slider*, el cual sirve para elegir el grosor del borde. Por ultimo esta *ToolButtons,* Estos botones se encargan de seleccionar que figura hacer y luego tambien para modificar las figuras ya creadas, por ejemplo dividirlas o duplicarlas. Todos los botones se extienden de clases ya existentes de la libraria javaFX.

Implementacion completa de la clase *PaintPane*

Esta es la clase mas importante del forntend ya que implementa todo lo necesario para unificar todo el programa. Esta gestiona tanto la interfaz de usuario como las interacciones del usuario con el lienzo, permitiendo dibujar y manipular figuras de manera intuitiva y organizada. Sus métodos y atributos están diseñados para ofrecer una experiencia de usuario completa y personalizable.

3. Problemas encontrados dureante el desarrollo

Comunicacion entre *PaintPane* y los botones

Uno de los primeros problemas encontrados durante el desarrollo fue que al modularizar y separar el comportamiento en bloques más chicos, muchas clases quedaron incomunicadas entre sí, esto se debio a que al principio implementamos muchas acciones dentro de la clase *PaintPane,* modularizando poco el codigo, y al querer corregirlo, por como fue planteado incialmente se complico, por lo que tuvimos que dejar el *LayerChoiceBoxButton* en *PaintPane*.

Distribucion de tareas

La primer dificultad que surgió fue la organización debido a los tiempos disponibles de cada integrante del grupo en época de finales, ya que la consigna del TPE fue dada una semana antes de la semana de finales. Esta superposición compromete la disponibilidad y energía de los integrantes del grupo, dificultando la gestión eficiente del proyecto. A pesar de estos obstáculos, logramos mantener una comunicación constante y una colaboración efectiva para abordar tanto las exigencias académicas como los requisitos del trabajo grupal, buscando estrategias para optimizar el tiempo y maximizar la eficacia en la consecución de nuestras metas.

En relación a la asignación de roles entre los integrantes del grupo, fue desafiante especificar tareas ya que nos parecía conveniente hacerlo de manera colaborativa y una comunicación constante entre todos los miembros. Por lo que la mayoria del codigo lo hicimos juntos a traves de *codeWithMe* que permitio que todos podamos hacer el trabajo a la vez y asi no tener problemas de merge. Aun asi cada integrante hizo cosas por su cuenta.

Dibujo de Figuras

Otro problema surgió al momento de dibujar las figuras. En la implementación original, se utilizaban múltiples condicionales para determinar el tipo de figura seleccionada y, dependiendo de ello, dibujarla de la manera adecuada. Para aprovechar mejor el paradigma de la programación orientada a objetos, se creó una nueva clase abstracta llamada *RenderFigure*. Luego, se desarrollaron clases específicas para cada tipo de figura dibujable, las cuales extienden de *RenderFigure* y contienen una referencia a la figura correspondiente del backend. Esto permite acceder a los datos de coordenadas necesarios para realizar el dibujo correctamente.