Diseño y Programación Orientados a Objetos
1er. Semestre 2020
Tarea 2: Aplicación Gráfica para Dron y su Control Remoto

Lea detenidamente la tarea. Si algo no lo entiende, consulte. Si es preciso, se incorporarán aclaraciones al final.
Objetivos de la tarea:

- Manejar proyectos vía GIT.
- Crear Interfaces gráficas en JavaFX.
- Incorporar medios auditivos a una aplicación Java.
- Incorporar menús.
- Aplicar animación con JavaFX.
- Aplicar “programación dirigida por eventos”.
- Ejercitar la creación y extensión de clases para satisfacer nuevos requerimientos.
- Generar documentación usando Javadoc.
- Identificar el patrón de diseño "Modelo-Vista-Controlador".

1.- Descripción General

En esta tarea se pide desarrollar interfaces gráficas para el manejo de un dron desde un control remoto como se mostró en la Tarea 1.
La funcionalidad del control remoto y dron en esta tarea es la misma de la tarea 1. Por favor, revise esa descripción. Se espera que su tarea muestre una interfaces similares a la Figura 1. La parte inferior muestra la vista del control remoto. La vista del dron es como si el usuario lo mirara de abajo.

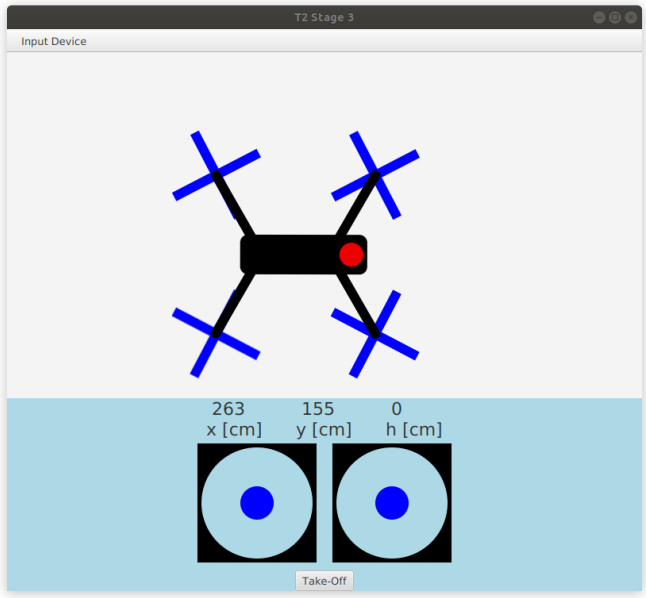

Figura 1a: Ejemplo de interfaz con joystikcs como dispositivo de entrada

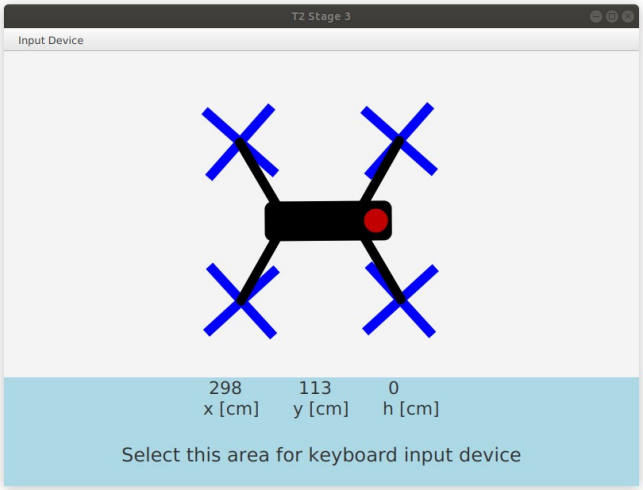

Figura 1b: Ejemplo de interfaz con keyboard como dispositivo de entrada

El menú superior ofrece dos opciones para el dispositivo de entrada: joysticks o keyboard. Usted puede elegir con qué dispositivo de entrada comienza su programa. Ante un cambio de dispositivo de entrada, no se requiere que el estado del dron sea informado al dispositivo de entrada (esto es opcional). Así, si el dron parte con keyboard y usuario cambia el dispositivo de entrada durante el vuelo, el control remoto podría mostrar Take-Off en el botón bajo los dos joysticks.
Se recomienda usar un BorderPane para disponer el menú en la parte superior, la vista del control remoto en la parte inferior y el espacio de vuelo del dron en el centro.
Para simular la altura del dron, éste reducirá su tamaño en la medida que sube. Calcule esta reducción para que a 5 [m] el dron se vea de la mitad de su tamaño a altura cero. La geometría de las

interfaces gráficas hace crecer el eje y hacia abajo. Se debe poner cuidado al expresar el movimiento del dron al usar esos ejes de coordenadas. Si bien los ángulos para determinar proyecciones con funciones trigonométricas se miden en radianes, JavaFX ocupa grados para al fijar la rotación de una figura geométrica. Ángulo cero se produce cuando hay coincidencia con el eje x. Por dirección del eje y, los ángulos crecen en dirección de las manecillas del reloj.
La unidad de los ejes de los “cristales” (Panes) de JavaFX es pixeles. En la a Tarea 1, se usó metros y para las velocidades metros por segundo. Parra no complicar esta tarea, se pide usar centímetros como unidad de medida. Así un centímetro será 1 pixel.
Esta tarea da libertad para que su grupo dé la forma que mejor estime al dron. Debe ser notorio qué parte del dron es la delantera. En Figura 1, se distingue por fuselaje desplazado hacia adelante y por una luz roja en la “cabina”. Se recomienda crear un dron grande (al menos 100 [cm]) para que se aprecie en pantallas que pueden ser de 1920x1080 pixles. El dron debe desplazarse, subir, bajar y rotar como se especificó en la Tarea 1.
Para aplicar animaciones las hélices del dron deben girar mientra esté en vuelo.
Para aplicar medios auditivos, el dron debe emitir sonido mientras vuela. Su grupo es libre para elegir el sonido o la música que acompaña el vuelo. Se recomienda un segmento corto que se reproduzca cíclicamente. Si el usuario presiona el botón derecho en la zona de vuelo del dron, un menú popup debe permitir cambiar la rapidez máxima de despegue/aterrizaje (ambas son iguales).

Una gran (inmensa) diferencia con la Tarea 1 la genera el nuevo paradigma usado en Tarea 2: “programación conducida por eventos”. En la Tarea 1, el método start activaba todos los cambios del programa. En esta tarea los eventos del usuario y los que usted programe gobernarán la ejecución del programa. En su solución trate que el programa reproduzca lo que ocurre en la realidad. Por ejemplo, existe un control remoto que lo vemos compuesto de dos partes: el dispositivo de entrada (joysticks y botón, o teclado) y el radio control que de manera inalámbrica envía órdenes al dron. Es por eso que solo el radio control contacta al dron, nunca el dispositivo de entrada. Un ejemplo de secuencia de eventos de entrada es: el usuario arrastra un joystick, el evento asociado calcula la fracción de desplazamiento vertical y horizontal y la reporta al control de la radio, éste envía las posiciones al dron. El dron, que revisa estos valores periódicamente (se pide cada 100 [ms] en tarea 1 y eso se mantiene aquí) y actualiza su posición, altura y rotación, y luego actualiza su vista.
Revise al final qué significa la arquitectura Modelo-Vista-Controlador, permite entender por qué es conveniente separar la vista de un objeto de su modelo. Esto no es estrictamente necesario cuando la vista es símple.
En Figura 1, usted puede notar que se si bien toda la parte inferior es la vista del control remoto, se distingue la vista de la telemetría recibida por el radio control de la vista del dispositivo de entrada. Como telemetría basta con actualizar cada 1 segundo las posiciones x, y, más la altura del drone. Revise la clase Timeline para programar tareas repetitivas. La vista del dispositivo de entrada debe cambiar cuando el usuario así lo decida desde el menú.
El dispositivo de entrada teclado (keyboard) requiere una interfaz visual, como la zona inferior mostrada en Figura 1b, para permitir que ésta tome el foco (setFocusTraversable(true)) y así poder recibir y manejar los eventos de teclado.
El dispositivo de entrada joysticks se muestra aquí con las vistas de dos instancias de joystick y un botón inferior. Arrastrando (dragging) con el mouse el centro de cada joystick hacia la periferia del círculo mayor se simula el movimiento de la palanca. El centro del círculo interior (azul en este caso) no debe salir del círculo mayor. Al soltar el botón del mouse, el círculo interior vuelve a su centro. Mientras se arrastra el mouse, este dispositivo de entrada envía la posición horizontal y vertical que tenga en la fracción que corresponda al radio del círculo mayor. Recuerde que los valores entregados por los dispositivos de entrada varían entre -1 y 1.

2.- Desarrollo en Etapas

Para llegar al resultado final de esta tarea usted aplicará nuevamente la metodología "Iterativa e Incremental" para desarrollo de software. Su grupo irá desarrollando etapas que irán abordando los requerimientos gradualmente. En cada etapa usted obtendrá una solución que funciona para un

subconjunto de los requerimientos finales. **Su grupo deberá entregar una solución para cada una de las etapas** aún cuando la última integre las primeras. **El readme y archivo de documentación deben ser preparados solo para la última etapa. Prepare un makefile para cada una de las etapas.** Esto tiene por finalidad, educar en la metodología iterativa e incremental.

2.1.- Primera Etapa: Crear vista para el dron
Cree una vista para su clase Drone de la Tarea 1. Como toda iteración de software, usted puede (debe) modificar su clase Drone de la Tarea 1 para mejor cumplir su propósito en esta tarea.
Recuerde que la vista que usted dé será la de dirección cero para la referencia con que JavaFX hace las rotaciones. Para esto, asegúrese que el frente del dron apunte hacia la derecha de la pantalla, misma dirección del eje x.
Para probar su vista, el código principal solo muestra una vista del dron con sus hélices girando. El programa termina al cerrar la ventana.

Como **ayuda sugerida**, usted puede analizar y completar (modificar, corregir) este código de partida para la vista DroneView de un dron cuyo modelo (y el control de sus eventos) se encuentra de la clase Drone. Vea el diagrama de clases para código de ayuda.

2.2.- Segunda Etapa: Control remoto sin dispositivo de entrada vuela el dron
Cree una vista para la radio del control remoto. La vista de la radio mostrará la telemetría del dron justo debajo de la vista del espacio de vuelo del dron. Los métodos del control remoto serán invocados desde Stage2Test para que el control remoto. El método start ordena despegar y girar a la derecha mientra avanza y sube (seguirá mientras el programa no termine). El programa termina 1 cuando el usuario cierra la aplicación. Verifique que el dron reduce su tamaño mientras sube. Use el programa de prueba Stage2Test.java proporcionado.

Como **ayuda sugerida**, usted puede analizar y completar (modificar, corregir) este código para la esta etapa. Vea el diagrama de clases para el código de ayuda.

2.3.- Tercera Etapa: Control remoto con joysticks hace volar el dron
Cree la vista para **un** joystick y luego la vista para el control remoto con **dos** joysticks. Haga un diseño modular para que luego pueda intercambiar el dispositivo de entrada entre joysticks y keyboard.
En su clase Stage3Test usted configura todo para que pueda controlar el dron a través de la interfaz gráfica con dos joysticks y un botón.

Como **ayuda sugerida**, usted puede analizar y completar (modificar, corregir) este código. Vea el diagrama de clases.

2.4.- Cuarta Etapa: Dron sonoro de rapidez de despegue/aterrizaje configurable controlado vía joysticks o keyboard según selección en menú
En esta etapa se espera lograr la funcionalidad completa del dron según la descripción de la primera parte de esta tarea.

Elementos a considerar en su documentación
Usted deberá documentar, usando notación "JavaDoc" las clases Joystick y JoystickView de su última etapa.
Prepare un archivo makefile para compilar y ejecutar su tarea en aragorn. Además incluya rótulos "clean" para borrar todos los .class generados, y "doc" para generar la documentación en directorio "documentation".
Entregue todo lo indicado en Normas de Entrega de Tareas. Su documentación automática con javadoc debe ser generable con:
$ make doc

Para que esto funcione usted debe incluir el rótulo doc en su archivo makefile.
(OJO en su entrega no incluya las páginas html generadas por javadoc, éstas serán generadas por este comando cuando el ayudante revise su trabajo)
En su archivo de documentación (pdf o html) incorpore el diagrama de clases de la aplicación (etapa 4). Éste lo puede generar con jgrasp u otro programa.

3.- Etapa extra crédito: Regreso al punto de partida con botón home y diagrama de la trayectoria

Su grupo puede aspirar hasta 7 puntos adicionales (la nota igual se satura en 100%) si desarrolla esta etapa además de la cuarta. Desde la Tarea 1 el dron ya registra su trayectoria en un archivo de salida. Se pide agregar un botón “Home” a la interfaz de Joysticks y responder al presionar “h” en el caso de entrada vía teclado. Ante esta orden y no ingresando nuevas órdenes el usuario, el dron sube a 500 [cm] si su altura es inferior a esa, gira para apuntar el puto de partida, vuela hasta el punto de partida, y baja hasta aterrizar. Una vez aterrizado, el usuario lo puede hacer volar nuevamente.

4.- Sobre la arquitectura Modelo Vista Controlador

Para organizar interfaces gráficas una "solución de software general recomendada" (éstas son conocidas como patrones de diseño) es el patrón "modelo-vista-controlador".
El modelo es la clase que caracteriza a un objeto y almacena los datos significativos de éste. Por ejemplo, en este caso la clase Drone maneja el modelo de un dron. Por otro lado tenemos las vistas, estas clases indican cómo un objeto se muestra visualmente. En nuestro caso se pueden identificar al menos dos vistas para el control remoto del dron, uno que considera joysticks y otro en que el teclado es el dispositivo de entrada. Para un dron, existe una única vista en esta tarea DroneView.
En otros problemas puede ocurrir que un mismo modelo tenga varias vistas. Por ejemplo un objeto termómetro tiene un modelo y podría mostrarse como columna de mercurio, como número digital, como una intensidad de color, etc.
Finalmente tenemos el controlador del objeto gráfico (se asemeja al control remoto de un dron). Las clases controladoras son aquellas que modifican los datos, por ejemplo a través de las acciones del usuario en la interfaz. Generalmente las clases controladoras corresponden a los "handlers" o “listeners”, es decir, los manejadores de los eventos que usted estima de interés. En esta tarea las clases anónimas (ocultas por el uso de expresiones lambda por ejemplo) genera cambios en el estado del dron (sus velocidades). Una clase puede cumplir dos roles, por ejemplo ser modelo y vista, como podría ocurrir en algunas clases de su tarea. Así como podemos tener varias vistas, es posible tener varias clases controladoras de un modelo. Recordar que detrás de cada expresión labda hay una clase anónima. Usted puede identificar clases controladoras observando la clase de los objetos que atienen los eventos de la interfaz (teclado y mouse).