Documentación
Tarea 3 - ELO329: Simulador Gráfico en C++ de la Evolución de una Pandemia. (Versión 1.1)

Escrito por: **Miguel Andrade**

Manuel Cruces

Kevin Pizarro

Pablo Troncoso

Julio 22, 2021

Explicación breve de la solución:

Se nos solicita implementar una interfaz gráfica de usuario (GUI) para la evolución de una pandemia. La GUI permite recibir parámetros iniciales a través de un archivo de configuración en formato txt y luego (algunas de ellas) pueden ser modificadas dentro de una ventana del simulador llamada **settings**. Por otro lado, la interfaz permite iniciar, pausar, acelerar y ralentizar la simulación a gusto. Por último, para poder interpretar y visualizar fácilmente la evolución de la pandemia se provee una sección para un gráfico de áreas apiladas, el cual se actualiza en tiempo "real". Para esto (utilizando el código de ayuda), se tienen 5 clases y 15 archivos para el proyecto, estos son:

- Clase: **Comuna**.
    - Archivos: **Comuna.h** y **Comuna.cpp.**
- Clase: **Pedestrian**.
    - Archivos: **Pedestrian.h** y **Pedestrian.cpp.**
- Clase: **Simulator**
    - Archivos: **Simulator.h** y **Simulator.cpp**
- Clase: **MainWindow**.
    - Archivos: **mainwindow.h**, **mainwindow.cpp** y **mainwindow.ui.**
- Clase: **Settings**.
    - Archivos: **settings.h**, **settings.cpp** y **settings.ui**.
- Archivo main: **StageX.cpp**
- Otros archivos: **StageX.pro** y **config.txt**

Extra.cpp es donde se encuentra el main y cuyo trabajo es inicializar simulador y comuna, así como recibir los parámetros del archivo de configuración (utilizando las librerías **iostream** y **fstream** para leer el archivo de configuración.

Estos parámetros se utilizan para crear la simulación, que es manejada por la clase Simulator, la cual maneja los tiempos de simulación, los métodos de simulación, la actualización del estado de los individuos y de solicitar a la comuna la información para generar el gráfico de áreas apiladas.

Para la ventana principal, y los menús de la misma, se encarga la clase mainwindow, la cual maneja toda la parte gráfica del programa, y tiene los menús que permiten iniciar o detener la simulación y abrir la ventana de settings, la clase mainwindow también crea y modifica el gráfico de áreas apiladas, permitiendo que se muestre en tiempo real, además de reiniciarlo cuando se pulsa Start. Además en la etapa extra la clase mainwindow se encarga de recibir las pulsaciones de las flechas para acelerar o desacelerar la simulación, lo cual se recibe mediante un

eventFilter, el cual se encarga de llamar a los métodos necesarios según sea el caso.

En la clase comuna se inicializa el “espacio” y arreglo de los individuos que se mueven en este, llamando a las actualizaciones de estado de los mismos según lo indique el simulador, además de enviar los datos de los estados de los mismos al simulador, cuando los solicita por parte de mainwindow. Luego se definen los vacunatorios y el método para ubicarlos en el espacio de la comuna. En Pedestrian se parametriza el comportamiento y estados de estos; si está infectado, vacunado, recuperado o es susceptible a infectarse. Luego se tienen los cálculos propios de su movimiento “aleatorio”, el cual está determinado por los parámetros entregados por el archivo de configuración.

Se adjunta el diagrama de clases UML.

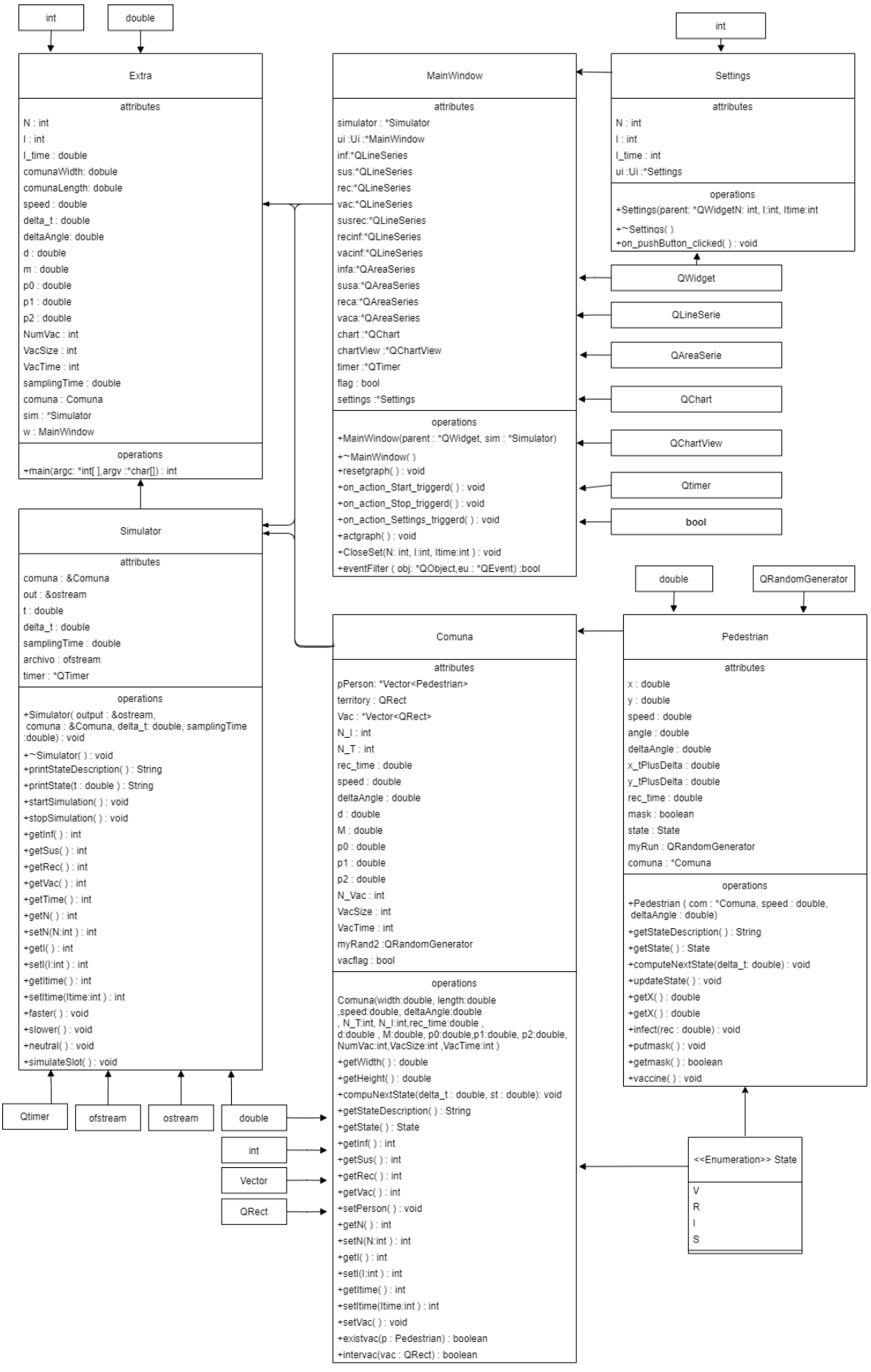

Figura 1. Diagrama de clases UML para stage Extra.

Dificultades encontradas:

1. Creación del menú settings.

Solución: Se creó otra form de QtCreator creando la clase Settings, usando la plantilla QDialog, la cual tiene como parámetros los I,N e I_time, para asignarlos a variables internas de la clase Settings las cuales al pulsar el botón de Ok, son actualizadas, y al pulsar Start, la ventana principal obtiene los valores de dichas variables para la simulación.

2. Creación correcta del gráfico

Solución: El gráfico de área de Qt, no es un gráfico de áreas apiladas, por lo tanto para poder simular uno, se “sumaron” las distintas series de datos, permitiendo que se muestre el gráfico como un gráfico de áreas apiladas.

3. Frecuencia de actualización del gráfico

Solución: Se creó en la clase mainwindow un Qtimer el cual tiene un tiempo de 200[ms] de actualización, y se vinculó la señal timeout del mismo con el método que actualiza el gráfico lo que permite que se actualice 5 veces por segundo, aunque esto no es muy obvio, ya que recibe los datos de la simulación cada un segundo. Se decidió no actualizarlo al tiempo que la simulación envía los datos, ya que en la etapa Extra se puede acelerar y desacelerar la velocidad de la simulación, y en el caso de que se acelere demasiado, el gráfico no sería visible.

Gráficos de Salida:

Para obtener el siguiente gráfico se simuló con la configuración dada por defecto en el repositorio de GitLab.

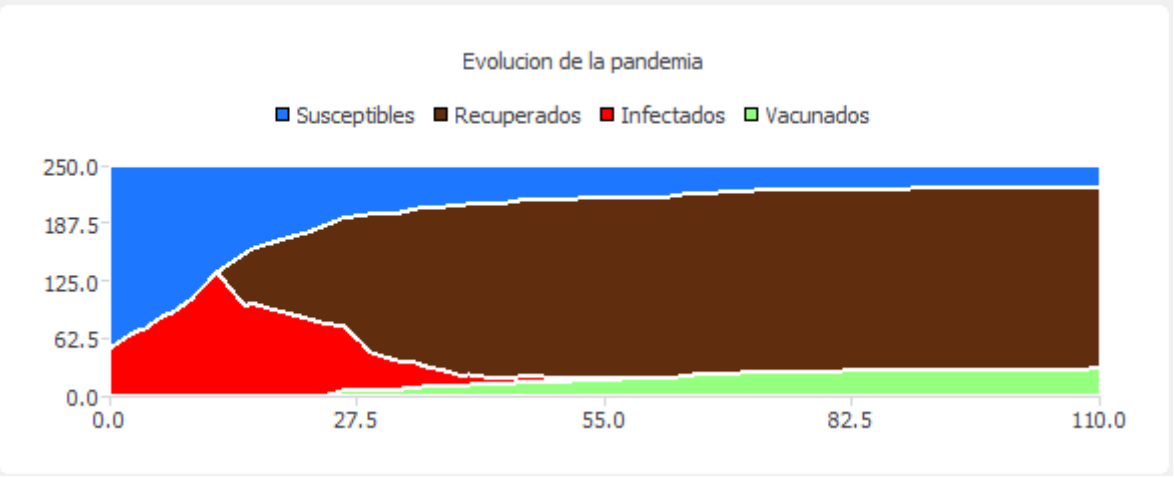

Gráfico 1. Salida de la etapa Extra.