(75.41) Algoritmos y Programación II

Cátedra Lic. Andrés Juárez

2do cuatrimestre de 2019

Manual de interfaz

Interfaz gráfica del Trabajo Práctico de clase: NANOBOT

Carolina Pistillo

1. Diagrama de clases

En la Figura 1 puede observarse un diagrama de las principales clases provistas en el código dado en el archivo *tpNanobot.zip*. En líneas generales, la clase *Juego* se encarga de la lógica básica del juego corriendo: un ciclo infinito de actualizar-renderizar. Por otra parte la clase *Entorno* se ocupa a grandes rasgos de renderizar imágenes utilizando la librería estándar de C/C+: *SDL*.

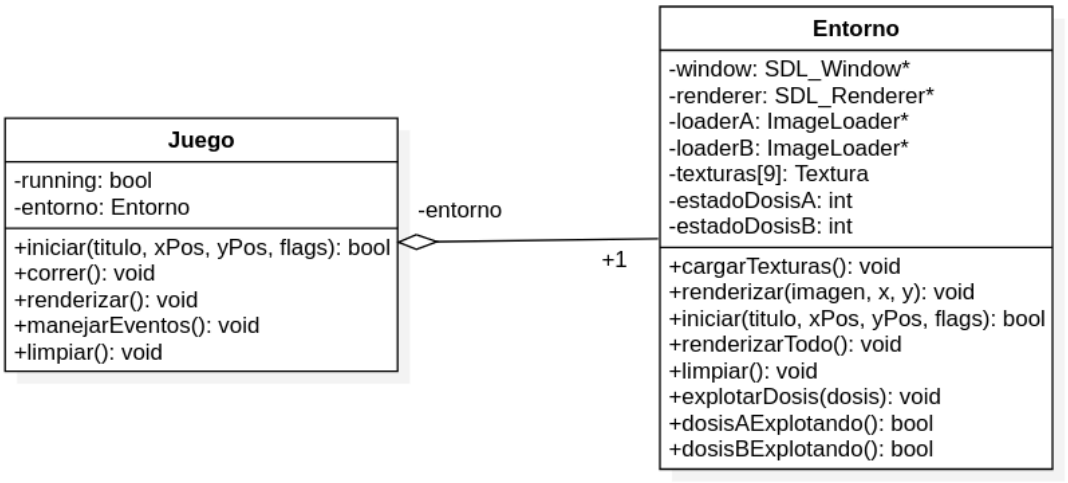

Figura 1: Diagrama de clases principales

1.1. Clase Juego

En particular la clase *Juego* tiene el método *correr()* que se encarga de mantener un ciclo infinito mientras el juego este corriendo (*running* es "true"). Dentro de ese ciclo podrá observar en el código que se envía un mensaje a la clase *FPSManager* para que comience y al final se le indica que finalice. El comportamiento de esta clase no resulta de nuestro interés para este trabajo, simplemente se encarga de manejar el comportamiento de los Frames Per Second que habrá en el juego.

El método más importante y que caracteriza a la clase en cuestión será entonces *correr()*, que mantiene el ciclo actualizar-renderizar a traves de los métodos *manejarEventos()* y *renderizar()*.

El método que maneja eventos se encarga de obtener dichos eventos por teclado a través de otra clase de utilidad llamada *InputManager*. Nuevamente esta clase no se incluyó en el diagrama porque no es de nuestro interés conocer su implementación. Sin embargo es importante la utilización del método *isKeyDown(KEY)* de la misma. Este método nos permite saber si la tecla KEY esta siendo presionada o no. Las constantes correspondientes a las teclas se hayan en el archivo *InputTable.h*.

Por otro lado, puede observarse que la responsabilidad del método *renderizar()* es delegada a la clase *Entorno* (que se ha colocado como atributo). El comportamiento de dicha clase se explicará a continuación.

1.2. Clase Entorno

Como se ha mencionado antes, la clase Entorno es la responsable de la interfaz gráfica del juego. Posee punteros a objetos de tipo *SDL_Window* y *SDL_Renderer*, que representan objetos de la librería gráfica correspondientes a la ventana donde se graficará y al objeto renderizador. Los atributos de tipo *ImageLoader** son instancias de una clase que se encarga de realizar cargas especiales de imágenes. En particular el *loaderA* cargará 20 imágenes correspondientes a la explosión de una dosis en sus colores originales. *loaderB* hará lo mismo pero con colores modificados. Los enteros de estado de las dosis indican el número de la imagen actual en que se encuentra la dosis (de 1 a 20). Si la dosis aún no ha explotado su estado será igual a 1.

Se presenta también como atributo un arreglo de objetos de tipo *Textura*, que representan objetos de las imágenes a cargar y renderizar, con ciertos comportamientos que no son de importancia en este contexto. Este arreglo puede accederse mediante el enumerado provisto en el código y contiene el fondo y los distintos microorganismos a representar gráficamente.

El método *iniciar(...)* se encarga de iniciar el entorno en un contexto de *SDL* y el método *limpiar()* se encarga de liberar los recursos utilizados por las texturas y objetos de *SDL*. *cargarTexturas()* se encarga de cargar las imágenes a utilizar desde los archivos correspondientes. La operación *renderizar(imagen,x,y)* realiza la lógica necesaria para renderizar una imagen del enumerado *imagenes* en la posición (x,y). Puede observarse que la responsabilidad de renderizar es delegada a la textura.

Uno de los métodos más importantes a tener en cuenta es *renderizarTodo()*. Realiza la lógica necesaria para preparar el renderizado y se encarga de renderizar las imágenes de fondo y al nanobot en determinadas posiciones. Dicho código solo debe modificarse donde se indica, es decir, agregar el código necesario entre medio de los comentarios especificados. Dentro de este método se deben realizar las renderizaciones necesarias con el método *renderizar(...)*. Si ya se esta trabajando con las uniones entre células recuerde que el código es secuencial y lo que se grafica primero irá más atrás que lo que se grafica luego. Por lo tanto, si desea graficar las líneas, debe hacerlo antes de renderizar otras imágenes. Una función útil para graficar líneas es *SDL_RenderDrawLine*.

El método *explotarDosis(dosis)* se encarga de cargar los sprites subsiguientes que simulan la explosión de una dosis e incrementan el estado de la misma. Los booleanos de dosis explotando serán de utilidad para el manejador de eventos dentro del juego.

2. Modificaciones y agregado de código

En primer lugar es interesante notar que el código dentro de la clase *Entorno* se halla poco modularizado y se le atribuyen responsabilidades que no se corresponden con el comportamiento que debería tener un entorno o interfaz. Esto fue hecho así debido a la imposibilidad de delegar responsabilidades a clases que todavía no fueron creadas. Por lo tanto, tenga en cuenta los comportamientos que se pueden delegar a la hora de crear más clases.

El agregado de más clases podría implicar además una mejora en la disposición de atributos como el arreglo de Texturas. Tenga libertad en modificar estas cuestiones.

Dentro de la clase *Juego*, es importante notar que no se tiene un método *actualizar()* dentro del ciclo de juego. Es importante tener en claro que en caso de ser necesario debe incluirse en el ciclo, siempre y cuando se encargue de actualizar cuestiones del juego que no tengan relación con el manejo de eventos por teclado. En esta primera aproximación del trabajo no fue necesario y por eso no se incluyó.

Como ha podido notar, debido a la dificultad que presenta y a modo de ejemplo de uso del manejador de eventos, se ha incluido en el código toda la lógica necesaria para la renderización de la explosión de una dosis al presionar las teclas A o B. Compruébelo usted mismo.

En el archivo *Constants.h* se presentan constantes que pueden ser útiles a lo largo del trabajo y que fueron utilizadas en ciertas porciones de código. Por último, tenga en cuenta no realizar cambios sobre las clases que no se presentan en el diagrama dado.