Proyecto de Programación

To Bee Or Not To Bee

Grupo: ToBeeOrNotToBee
Bhavuk Sikka

- **Descripción General**

En un futuro lejano, los insectos vuelven a ser grandes y con ello ganan gran inteligenica e influencia en su entorno. Llegan a tener tanta relevancia como los humanos en la actualidad.

El jugador se encarna en la abeja reina de una colmena futurista y ha tomar decisiones para su reino, que afectarán en gran medida el estado de la corona. Dicho estado se representa mediante cuatro indicadores que se muestran a la derecha de la pantalla, que son: La comida, que representa la cantidad de alimento que tiene la colmena para sobrevivir; el pueblo, que es la aceptación de la reina por parte de las masas; el armamento, que indica cuán protegido está la corona con la milicia; y por último el dinero, el cual tiene la misma importancia que en la vida real, siendo imprescindible para realizar cambios en el reinado.

Esto dota al juego de una dificultad media, sin hacerlo insanamente difícil, ya que elegiendo apropiadamente se puede sobrevivir, pero tampoco facilitándolo tanto tal que elegiendo aleatoriamente en las decisiones esté siempre vivo. Nótese que no se puede ganar el juego, sino solamente mantenerse vivo por el máximo tiempo posible, es decir, la reina está condenada a la muerte desde que empieza a jugar.

Además, las decisiones a tomar se generan aleatoriamente al principio de cada partida. No obstante, como hay un número limitado de decisiones a tomar, si el jugador completa todas estas decisiones sin morir, se genera una nueva permutación entre las mismas. Es decir, se juega el mismo contenido una y otra vez, hasta morir finalmente.

El juego no tiene ningún modo de guardar la partida, ya que carece de sentido en este caso al consistir en no morir por el mayor tiempo posible. No obstante, como calificador de cómo se ha jugado la partida se tiene una puntuación final tras cada partida.

Es preciso mencionar la estructura del código a grandes rasgos. Hemos optado por hacer una función "main" bastante modesta, manejando únicamente la inicialización de la música y futuros sonidos que se usan en el juego mediante hilos así como el acceso a los distintas entradas (el juego como tal, un mini tutorial, los ajustes, los logros y salir del juego).

- **Aspectos Técnicos**

El juego lo hemos desarrollado partiendo de cero, a excepción de la música. Para implementar sonidos básicos teníamos a disposición una API bastante intuitiva de la librería FMOD. Cabe mencionar que para músicas de larga duración, ergo más pesadas, teníamos que esperar bastante tiempo (unos segundos). Por ello, decidimos poner una pantalla de carga al inicio del juego.

Para la parte gráfica se ha creado una librería llamada grafiks.c/h, que contiene todo lo esencial para representar la información en la pantalla de manera ordenada. Se ha seguido una estructura de puntos y secciones. Gracias a esto se han definido tres secciones principales, para mostrar todo el texto, para dibujar la abeja y para mostrar los indicadores del estado de la colmena. Para más detalles, véase **Mi Aportación**.

Como la interacción del jugador es escasa, decidimos hacer una interfaz intuitiva con un indicador que está siempre seleccionando la opción actual. El jugador puede moverse por las múltiples opciones mediante las flechas del teclado, y elegirla dando a la tecla *Enter*.

La lógica del juego consta de una serie de "*quests*", que son las decisiones que ha de tomar el jugador. Tenemos un archivo de configuración donde se indica el número total de *quests*, que es 50. En base a esto se construye una permutación aleatoria de números del 1 al 50, la cual se lee para saber cuál *quest* mostrar en cada momento. No obstante, para dar mayor emoción al juego incluímos posibles continuaciones a cada decisión, por lo que tomada una decisión podría ocurrir en

el futuro algún suceso relacionado. Por ejemplo, si se decide ignorar un peligro mayor, hay una pequeña posibilidad de que todo el reino se derrumbre por ignorarlo. Dicho comportamiento se ha logrado utilizando una implementación de colas hecha de otros trabajos. Esta cola contiene las continuaciones, que se van vaciando cada cierto tiempo.

- **Dificultades Imprevistas**

  No sería un proyecto en el que haya aprendido algo si no hubiera habido dificultades.

  Al inicio del proyecto tenía en mente hacer todos los diseños de abejas y texto únicamente con una función plot. Esta función se ocuparía de dibujar un polinomio de hasta grado 4 en la pantalla, contenido en una sección. No obstante, tras darle muchas vueltas lo vi muy complicado para la utilidad que iba a tener, pudiendo causar más problemas que solucionarlos. Al final esta dificultad la superé ignorando esta función y me ceñí con lo importante. Lo cual me lleva al diseño de las abejas, en las que tuve que invertir la mayor parte del tiempo puesto que necesitaba poder crear nuevas apariencias de abejas de manera razoonable.

  Otro gran problema con el que nos topamos fue el manejo de hilos y los semáforos. Al no saber cómo usar la librería pthread, fuimos postergando esta parte del trabajo más y más hasta que no tuvimos más oportunidad que enfrentarnos a los hilos.

  No obstante, la dificultad imprevista más problemática fue casi llegar a perder un día de trabajo muy productivo, equivalente a una semana completa de trabajo. Tras mucha tensión y gracias a la ayuda de mi compañero Samuel, conseguí recuperar gran parte del código. De esto aprendí a no hacer "*git reset hard*", pudiendo hacer "*git restore algún_archivo*".

- **Organización Del Trabajo**

  Al ser nuestro primer proyecto serio, no hemos sabido organizarnos fenomenalmente. Se podría decir que trabajabamos en el juego cuando podíamos, aportando nuevas ideas y código al juego cada cierto tiempo. Por ejemplo, mi compañero Samuel veía que había una gran parte del juego que estaba sin hacer, como la lógica del juego, y se ponía a ello de inmediato.

  Esta organización anárquica condujo en parte a una aportación del proyecto desequilibrada, aunque al final se logró el objetivo en mente.

  Además, para comprobar todas las librerías que ibamos creando y corregir los posibles errores y pérdidas de memoria, íbamos creando muchos archivos prueba*librería*.c. De este modo, tendíamos a copiar el código medianamente bien estructurado en estos archivos de prueba para integrarlo en las funciones principales.

  Para facilitar la comunicación y escribir siempre en el código más actualizado utilizamos el GitLab que la Escuela Politécnica Superior de la Universidad Autónoma de Madrid nos proporciona. Sin embargo, al ser mi primera vez usando esta herramienta, fui algo torpe, llegando a ejecutar "*git reset hard*" como se explica en el apartado previo. Cabe mencionar que aprendí lo básico de esta herrmienta, y fue de mi agrado. Por ello, si tuviera que hacer otro proyecto, utilizaría git sin dudarlo, pudiendo incluso montarme un servidor de GitLab si fuera necesario, ya que entre las ventajas está el guardado de todas las instancias del proyecto y la facilidad de usarlo.

  Dada la situación excepcional de este año, la comunicación era imprescindible. La mayor parte de cambios al proyecto se veían reflejados en conversaciones, ya sean privadas o públicas del grupo. Gracias a todas las tecnologías que tenemos hoy día, no fue necesaria ninguna reunión presencial. Aunque cada clase que teníamos del proyecto nos reuníamos los que podíamos en un chat de voz, podríamos haber realizado todo el trabajo sin siquiera una reunión online; las conversaciones por chat de texto eran suficientes, conjunto con git.

- **Mi Aportación**

En mi caso, lo más relevante que aporté fue la librería *grafiks*, que principalmente sirve para mostrar texto, dibujar líneas e inicializar la terminal. La idea en la que se basa el código es tener la terminal dividida en múltiples secciones, que vienen definidas por un punto y el tamaño (el número de filas y columnas). Para mostrar el texto en pantalla, normalmente se debería de utilizar la función textbox, que muestra el texto en una sección con la alineación indicada. Por otro lado, para dibujar líneas se usa draw_line, que pinta una línea de un punto a otro dentro de una sección. De este modo, se pueden pasar puntos que estén fuera de la sección indicada, pero teniendo la línea siempre dentro de dicha sección. A pesar de que esta función esté diseñada para dibujar líneas, se ha utilizado incluso para escribir texto en una única línea. Ambas funciones, al necesitar mover el cursor e imprimir sobre pantalla, utilizan un semáforo (mutex) para que no haya varios hilos tratando de mostrar información en pantalla simultaneamente.

Adicionalmente, me ocupé del diseño de todas las abejas, que fue lo que más tiempo consumió. Para poder tener muchos diseños de abejas fácilmente programables e integrarlo con lo programado por mi compañero Samuel (la lógica del programa), cree archivos que contenía el aspecto de las abejas codificadas. Aunque en un principio pensaba agregarles ropa a las abejas, la falta de inspiración del diseño lo evitó.

Finalmente, colaboré con mi compañero previamente mencionado en varios módulos de código (archivos .c), corrigiendo bugs, añadiendo cambios sustanciales, como la pantalla de carga, y aportando ideas nuevas. Se puede ver en cada archivo .c/.h los autores de este.

- **Si Volviera Atrás...**

La idea detrás del juego estuvo basada en un juego ya existente en Android, llamado "*Lapse*". Aunque la idea me gustó, habría elegido algo más ambicioso, como un shooter en primera persona online, parecido al modo multijugador de "*Call of Duty*" en la terminal. Para lograr esto, estaríamos obligados a entender cómo establecer conecciones cliente a servidor, usando la librería socket.h entre otras. Es seguro que nos habría llevado mucho tiempo más, siendo poco razonable para el tiempo que teníamos.

El aprendizaje durante este proyecto ha sido constante, ya que cada semana aportaba nuevo código al proyecto tratando de equilibrar código donde tuviera que pensar de verdad, o diseñar, con el código repetitivo como getters, setters, error checking, etc. Pero aún había algo que faltaba, que es la posibilidad de actualizar el juego. Al leerse el número total de preguntas desde un archivo *config*, podríamos fácilmente implementar con git un método para actualizar estas cuestiones e incluso añadir nuevas. Simplemente tendríamos que subir los nuevos archivos a un repositorio público de git, cambiando también el archivo *config*. De este modo, el juego siempre trataría de hacer "*git pull*" con una llamada al sistema, y continuar la ejecución del resto del juego como si nada.

- **Vista General**

En mi experiencia, el proyecto no me ha supuesto grandes problemas ya que sabía cómo programar, aunque está claro que muchos aspectos del juego se podrían mejorar, como apariencias más bonitas y más variados, tener un scoreboard global, para la que podríamos utilizar la librería *odbc* para manejar bases de datos, etc.

Por ello, pienso que la mayoría de las clases no eran de gran utilidad en lo respecto a enseñanza, ya que todo lo esencial que teníamos que saber fue impartido en las primeras clases. A esto se le suma que las dudas las terminabamos buscando en Internet la mayor parte de las veces.

Sin embargo, las clases de los Lunes en las que presentábamos nuestro proyecto hasta el momento solían ser de gran ayuda, ya que establecían una fecha "límite", poniendo mayor presión al desarrollo del juego. Esto se tradujo en que la mayor parte del trabajo se condesara en los Lunes y los Jueves lectivos.

Algo que me gustaría que se cambie es el enfoque de la creación del juego. En lugar de pensar que estamos haciendo el juego simplemente porque el profesor nos lo pide, habría sido más emocionante si el juego se fuera a jugar por muchas personas. Como poder publicar todos los juegos en alguna plataforma de la UAM para que otros usuarios puedan jugarlo, o incluso realizar alguna exposición del juego al público. Esto fomentaría un pensamiento más perfeccionista, agregando detalles que dotarían al juego de mayor emoción y cariño.

En conclusión, proyectos como este fortalecen bastante todo lo ya aprendido y expanden los conocimientos, despertándo de nuevo la ilusión de programar.