Examen - Parte Práctica

03 de julio del 2018

1. Instrucciones

Tendrás 4 horas para desarrollar el problema propuesto, en forma individual. Puedes usar todo el material del curso y acceder a internet, sin embargo, está estrictamente prohibida la comunicación con otras personas a través de cualquier medio.

Se ha habilitado un espacio en los repositorios para subir tu trabajo, y al igual que para los laboratorios y el proyecto, solo allí se recibirán las entregas dentro del plazo indicado por el equipo docente.

El no cumplimiento de cualquiera de estas normas implicará la reprobación inmediata del examen, tanto en su parte práctica como teórica.

2. Introducción

El objetivo de esta actividad es construir una versión adaptada del clásico Pacman, donde el usuario controla el personaje amarillo intentando obtener la mayor cantidad de puntaje sin ser atrapado por los fantasmas.

En concreto, deberás implementar un juego donde existen dos fantasmas (el rojo y el rosado) que van a la caza de Pacman. Además, en forma aleatoria y por un tiempo limitado, pueden aparecer frutas que permiten ganar puntos a nuestro protagonista cuando se las come. Finalmente, y al igual que en el original, el juego termina cuando uno de los fantasmas toca a Pacman.

3. Requisitos

- En forma individual, deberás desarrollar una aplicación de Windows Forms o Gtk# que permita jugar un nivel de Pacman.
- Al comenzar, debes dar la bienvenida al usuario y solicitar su nombre.

- Pacman se ubicará al centro de la pantalla, y debe poder ser controlado con las flechas del teclado.
- Debes posicionar en forma aleatoria a los fantasmas Rojo y Rosado, y siempre deben perseguir a Pacman. Es decir, deben moverse hacia nuestro personaje principal sin intervención del usuario.
- Existen dos frutas, la guinda y la uva. La primera da 10 puntos a Pacman y la segunda, 15.
- Las frutas deben aparecer en pantalla cada 30 segundos, y desaparecer luego de 10 segundos si es que Pacman no se las ha comido.
- Si Pacman se come una fruta, debe aumentar su puntaje según el valor de la fruta. Luego, la fruta debe desaparecer de la pantalla.
- Es importante que Pacman y los fantasmas se mantengan dentro de la pantalla en todo momento, es decir, no deben salirse por los bordes.
- Tu programa debe permitir al usuario guardar su progreso (nombre y puntaje actual), y retomar el juego más adelante. Esto, a través de botones en la pantalla de juego o con un menú en la parte superior de la ventana.
- Finalmente, si uno de los fantasmas alcanza a Pacman, el juego debe terminar y mostrar en una ventana emergente el puntaje obtenido y un mensaje de derrota con el nombre de usuario.
- Deberás llevar una tabla de posiciones histórica. Luego del mensaje de derrota, debes mostrar en otra ventana los 10 mejores puntajes que han habido en el juego, la duración de cada una de esas partidas (en segundos), y el nombre del jugador que obtuvo el récord.
- También debe poder accederse a la tabla de posiciones desde la pantalla de bienvenida del juego, con un botón o con un menú en la parte superior de la ventana.
- Junto con la tabla de posiciones, deberás crear un archivo de texto con las estadísticas de la partida recién terminada: nombre de usuario, duración del juego (en segundos), puntaje obtenido y nombre del fantasma que mató a Pacman.
- En pos de la experiencia de usuario, tu programa no debe caerse y terminar en forma inesperada. Si se encuentra un error grave, debes informar sobre el error al usuario a través de una ventana emergente.
- En esta misma línea, podrás optar a puntaje adicional si agregas sonidos o imágenes adicionales a lo solicitado en este enunciado y que ayuden a generar una atmósfera de juego.
- Tu programa debe permitir programar nuevas frutas y fantasmas realizando la menor cantidad de cambios posibles. Indica con comentarios en el código cuáles serían las modificaciones a realizar para lograrlo.

4. Trucos y Tips

Una estrategia muy usada en la programación de videojuegos es contar con un ciclo que vaya chequeando en forma periódica los distintos sucesos y reglas que deben ocurrir en cada momento. La periodicidad se logra con un objeto `Timer`, que gatilla un evento en un periodo de tiempo definido. En Gtk#, un equivalente es el método `GLib.Timeout.Add(uint interval, new GLib.TimeoutHandler(método_a_invocar)`.

Es fundamental para la interacción del usuario poder capturar los eventos del teclado. Si bien en Windows Forms esto se logra trivialmente al suscribirse al `KeyPress` de la ventana, en Gtk# debemos poner el atributo `[GLib.ConnectBefore]` sobre el método que recibe ese evento. Si no lo hacemos, el `KeyPress` será respondido por el sistema y no podremos trabajar con el evento.

Otro elemento clave para el éxito del juego es la correcta detección de colisiones. Una estrategia que facilita esta tarea es usar la clase `Rectangle`, pues provee métodos para acceder a sus coordenadas y para chequear si existe intersección entre dos objetos de ese tipo.