PANG

Segundo trabajo de Programación de Sistemas

Iñaki García Valero, 22136
Steffen Pablo Otten, 22252
Javier Melero Moreno, 20216

Introducción

El objetivo de este trabajo es desarrollar, a partir del código proporcionado, un videojuego inspirado en el clásico PANG de Atari, utilizando el lenguaje de programación ANSI C++ y la biblioteca gráfica multiplataforma OpenGL.

El juego consiste en controlar un personaje que debe esquivar y disparar a las pelotas que aparecen en pantalla. Estas pelotas rebotan al impactar contra el suelo y, al ser disparadas, se dividen en dos pelotas más pequeñas, hasta un máximo de 2 veces.

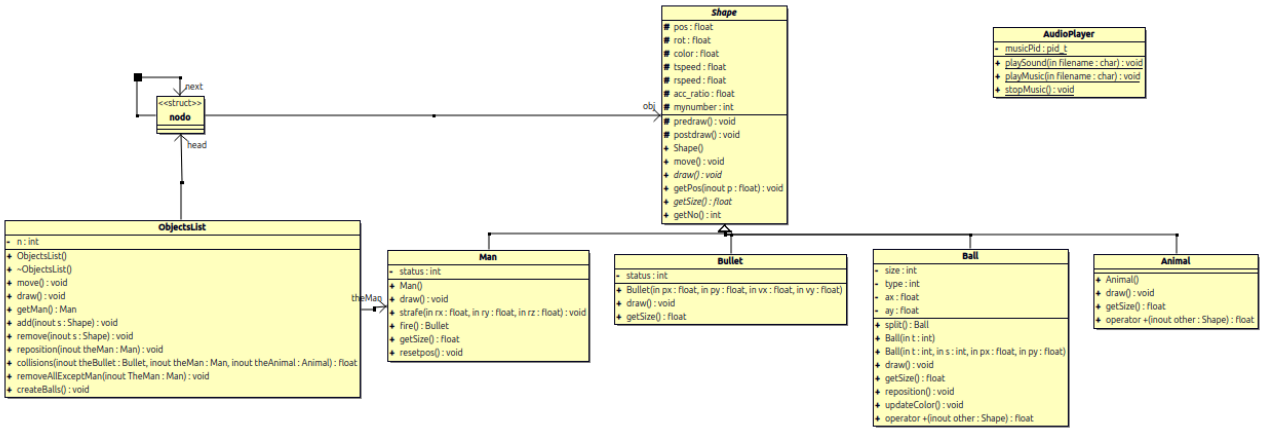
Además, se han implementado extensiones obligatorias, como la aparición de animales que recorren la pantalla y otorgan puntos extra, así como algunas extensiones opcionales, entre las que se incluyen una pantalla de inicio, de “Game Over” o un sistema de puntuaciones.

A continuación, se detallan los aspectos principales abordados durante el desarrollo del trabajo.

1. Diagrama de clases.

El diagrama de clases es una representación gráfica que muestra la estructura de un sistema, incluyendo las clases que lo componen, sus atributos, métodos, y las relaciones entre ellas. Este tipo de diagrama resulta especialmente útil para visualizar de manera clara y concisa la arquitectura del programa antes de su implementación.

El diagrama de clases implementado en este proyecto es el siguiente:



Como se puede apreciar, se compone de una clase principal denominada ***Shape***, de la cual heredan otras clases como ***Animal*** o ***Bullet***, entre otras.

Adicionalmente, se ha definido una estructura *nodo*, que contiene un puntero de tipo *Shape*, y otro puntero al siguiente nodo (*next*), formando así una lista enlazada.

Sobre esta base, se ha creado la clase ***ObjectList***, que gestiona la lista de objetos del juego. En ella, se instancia un nodo *head*, que apunta al primer elemento de la lista, y se implementan diversas funciones que permiten recorrer, modificar y administrar dicha lista.

Finalmente, la clase ***AudioPlayer***, que se encarga de la reproducción de audios, se gestiona de manera independiente al resto de clases y estructuras del sistema.

2. Descripción general de funciones y estructuras de datos más representativas.

A lo largo del desarrollo del proyecto se llevaron a cabo múltiples cambios sobre el código base, abarcando la implementación de nuevas clases, mejoras en las existentes y una reorganización general del funcionamiento del juego.

En la clase ***Ball***, se modificó el constructor para ajustar la generación de posición inicial y evitar que las pelotas se queden atascadas en los bordes. Se implementaron los métodos *updateColor()*, *getSpeed()*, *setSpeed()* y *setPos()* para controlar aspectos como color y rebote. El método *split()* fue reescrito para ajustar la velocidad de las pelotas hijas tras una colisión y actualizar el color, y también se implementó el *operador* + para calcular la distancia con otros objetos.

La clase ***Bullet*** fue modificada para cambiar el color de la bala a azul, lo cual mejora su visibilidad durante el juego. Por otro lado, se implementaron desde cero las clases ***Animal*** y ***Obstacle***, cada una con sus respectivos métodos de dibujo, colisión y gestión de atributos específicos. En *Obstacle* se incluyó el atributo *size[2]* y se añadieron funciones como *getDistanceToEdge()*, mientras que *Animal* fue diseñada para generarse aleatoriamente en la parte superior de la pantalla y desplazarse hacia abajo.

Se introdujo la clase ***ObjectList***, que gestiona todos los objetos activos del juego mediante una lista enlazada. En ella, además de los métodos requeridos en guión, se han añadido los métodos *removeAllExceptMan()*, *createObstacles()* y *createBalls()*. Esta clase también incluye punteros al jugador (*theMan*) y al primer nodo de la lista (*head*).

Por otro lado, se desarrolló la clase ***AudioPlayer***, encargada de la reproducción de música de fondo y efectos de sonido. Esta clase incorpora el atributo *musicPID* y métodos como *playSound()*, *playMusic()* y *stopMusic()*, los cuales gestionan procesos hijos para ejecutar los archivos de audio sin bloquear el hilo principal del juego.

En cuanto a la clase ***Shape***, se eliminó la aleatoriedad del rebote en el método *move()* para evitar errores en la física del juego. La clase ***Man*** fue actualizada para incluir sonido al disparar mediante la función *fire()*.

En el archivo principal del juego (***mainPANG***), se añadieron los headers de las nuevas clases y se reescribieron las funciones de entrada por teclado para mejorar la respuesta al movimiento del jugador. Se añadieron las funciones *setScore()* y *getHighScore()* para gestionar un sistema de

puntuaciones almacenadas en archivo de texto. Además, se incorporaron dos nuevos menús: uno de inicio y otro de *Game Over*, controlados mediante la variable *gameState*. Se agregaron las funciones *drawBox()*, *printText()*, *printMenu()* y *printGameOver()* para renderizar estos menús.

La función *myLogic()* fue actualizada para integrar completamente la detección y respuesta ante colisiones entre bolas, el jugador, proyectiles, animales y obstáculos. También se modificó la gestión de la bala, que ahora desaparece al salir de la pantalla, en lugar de depender de un timer; y se incluyó la generación de animales aleatorios con una probabilidad del 1%. Finalmente, se integraron efectos de sonido y música asociados a distintos eventos del juego, y se creó la función *resetGame()* para reiniciar correctamente todos los elementos en cada nueva partida.

Por último, el *Makefile* fue actualizado para compilar todos los nuevos archivos y asegurar el correcto enlace del ejecutable en entornos Linux.

3. Descripción de pruebas realizadas para probar el correcto funcionamiento del algoritmo.

El programa ha sido desarrollado principalmente en un entorno Linux Ubuntu, utilizando el compilador ***g++*** y la herramienta ***gdb*** para depuración. Para facilitar la cooperación entre los miembros del equipo, se ha utilizado la plataforma *GitHub* como sistema de control de versiones y repositorio del código.

Durante las pruebas en Ubuntu, el programa funciona correctamente, con un rendimiento y comportamiento coherentes con lo esperado. No obstante, hemos probado el programa en los escritorios virtuales de la UPM, donde se han observado velocidades de ejecución notablemente más altas, lo cual dificulta la jugabilidad y altera significativamente la experiencia.

Adicionalmente, se ha intentado ejecutar el programa en sistemas Windows mediante *Windows Subsystem for Linux* (WSL). Aunque el juego se ejecuta correctamente, también presenta problemas de velocidad similares a los mencionados anteriormente. Además, WSL no reconoce el directorio donde se encuentran los archivos de audio, impidiendo la reproducción de sonidos y música. Este problema parece ser propio de la aplicación WSL, y que no debería producirse en entornos Linux nativos.

Para entornos nativos de Windows, el programa podría ejecutarse correctamente siempre que se cuente con la instalación adecuada de la biblioteca gráfica OpenGL. Sin embargo, la funcionalidad de reproducción de audio presenta una limitación importante, ya que actualmente está implementada utilizando herramientas y librerías propias de entornos Linux. Por lo tanto, para lograr una ejecución completa en sistemas Windows, sería necesario adaptar o reemplazar esta parte del código con una solución compatible.

4. Guía de uso del programa ejecutable.

El programa debe ser compilado en un entorno Linux mediante el comando:

make Linux

Esto creará un ejecutable *PANG*, que al ejecutarlo abrirá el juego.

Una vez ejecutado el juego, se mostrará la pantalla inicial: un menú de bienvenida acompañado de música de fondo. En esta pantalla se indican las instrucciones básicas para comenzar a jugar. Para iniciar la partida, basta con pulsar la barra espaciadora del teclado. Durante el juego, el jugador puede desplazarse utilizando las flechas de dirección y disparar también con la barra espaciadora.

Al comenzar la partida, el menú de inicio desaparecerá y se cargarán los distintos elementos del juego: el personaje principal, las pelotas, los obstáculos, y de forma ocasional, un pájaro amarillo que atraviesa la pantalla verticalmente. En la parte superior de la pantalla se muestra la interfaz de usuario, donde se puede ver el número de vidas restantes, la puntuación actual y el récord histórico (*HighScore*).

La mecánica del juego es simple: evitar ser alcanzado por las pelotas mientras se dispara tanto a ellas como al pájaro para obtener puntos. Cada vez que una pelota es impactada, se divide en pelotas más pequeñas, lo que incrementa la dificultad a medida que avanza la partida.

Cuando el jugador pierde todas sus vidas, aparece la pantalla de "Game Over". En ella se muestra la puntuación final alcanzada, el *High Score* y la opción de reiniciar la partida pulsando nuevamente la barra espaciadora.

5. Descripción del reparto de roles del equipo.

Para desarrollar este trabajo, decidimos hacer entre todos la clase *ObjectList* antes de que empezara la época de exámenes. Cuando nos funcionó correctamente esta parte del código, decidimos empezar a realizar la parte obligatoria y adicional que requería el trabajo, de tal forma que Iñaki se encargó de hacer la parte del animal, y entre Steffen y Javier, se encargaron de hacer los obstáculos y el menú de inicio y Game Over. Como a Steffen se le ocurrió y tenía más tiempo, hizo la parte musical y sonora. Cabe destacar que él también fue de gran ayuda cuando el resto del equipo no sabíamos qué estábamos haciendo mal y nos ayudó a orientarnos en nuestras tareas. Para la realización de este documento nos involucramos todo el equipo.

6. Propuestas de mejora y valoración personal.

Consideramos que el programa desarrollado tiene un amplio margen de mejora y expansión. Existen numerosas ideas y conceptos adicionales que se podrían implementar para ofrecer un trabajo aún más complejo. Algunas de estas ideas son añadir bonificaciones por combos, diferentes tipos de animales con distintos efectos sobre el jugador, varios niveles de dificultad o incluso mejoras visuales como animaciones al impactar o al finalizar la partida.

No obstante, muchas de estas ideas implican un grado de complejidad considerable, por lo que no fueron implementadas en la versión final del proyecto.

Una mejora concreta que nos hubiera gustado integrar, pero que finalmente no fue priorizada, es la gestión de la condición de victoria del juego. Aunque el jeugo podría acabar una vez eliminadas todas las bolas de la pantalla, esto rara vez ocurre debido al alto nivel de dificultad. Dicho estado no está programado, y si se alcanza, el juego se quedaría atascado. Por este motivo, optamos por establecer un sistema de puntuación en el que el objetivo sea conseguir la mayor cantidad de puntos posibles antes de perder todas las vidas.

En general, estamos muy satisfechos con el desarrollo del trabajo. Hemos logrado los objetivos que se nos pedía, y también hemos logrado añadir detalles extra que mejoran el programa. El resultado final es un juego completo y entretenido, que tiene todo lo necesario para que funcione correctamente y resulte una experiencia divertida.