ESCUELA DE INGENIERÍA INFORMÁTICA

PONTIFICIA UNIVERSIDAD CATÓLICA DE VALPARAÍSO

Programación Avanzada

INF2236-2

Profesores:
Claudio Cubillos
Leonardo González

Integrantes:
Agustín Guzmán
Nicolás Leiva
Felipe Márquez

AVANCE

GM1.1 Explicación del juego (perspectiva del jugador)

Narrativa y objetivo: Eres un conductor en una autopista. Tu meta es recolectar dinero para sumar puntos mientras esquivas patrullas de policía. Cada choque con la policía te quita una vida. Cuando llegas a cero vidas, la partida termina y puedes volver a intentarlo.

Elementos del juego: El jugador controla un auto con vidas, puntos y un estado temporal de “herido” tras recibir daño. Caen objetos desde la parte superior de la pantalla: las patrullas de policía causan daño y el dinero otorga puntos y activa un sonido. La interfaz (HUD) muestra Dinero (puntos), Vidas y el HighScore. El juego tiene cuatro pantallas: Menú principal, Juego, Pausa y Game Over.

Controles: Flecha izquierda para mover a la izquierda, flecha derecha para mover a la derecha, tecla P para pausar y clic/toque para navegar en Menú y Game Over.

Maquetas/pantallas: En el Menú principal se muestra el título y un mensaje para tocar y comenzar. En la pantalla de Juego se ve el fondo de la carretera, el auto en la parte inferior y la caída de objetos (policías y dinero) con el HUD en la parte superior. En Pausa, se muestra un mensaje con la opción de continuar tocando la pantalla. En Game Over, se muestra el mensaje de fin y la opción de reiniciar tocando.

Recursos visuales y de audio: Fondo (assets/background.png), Auto (assets/car.png), Policía (assets/police.png) y Dinero (assets/cash.png).

GM1.2 Análisis del juego (perspectiva de Ingeniería de Software)

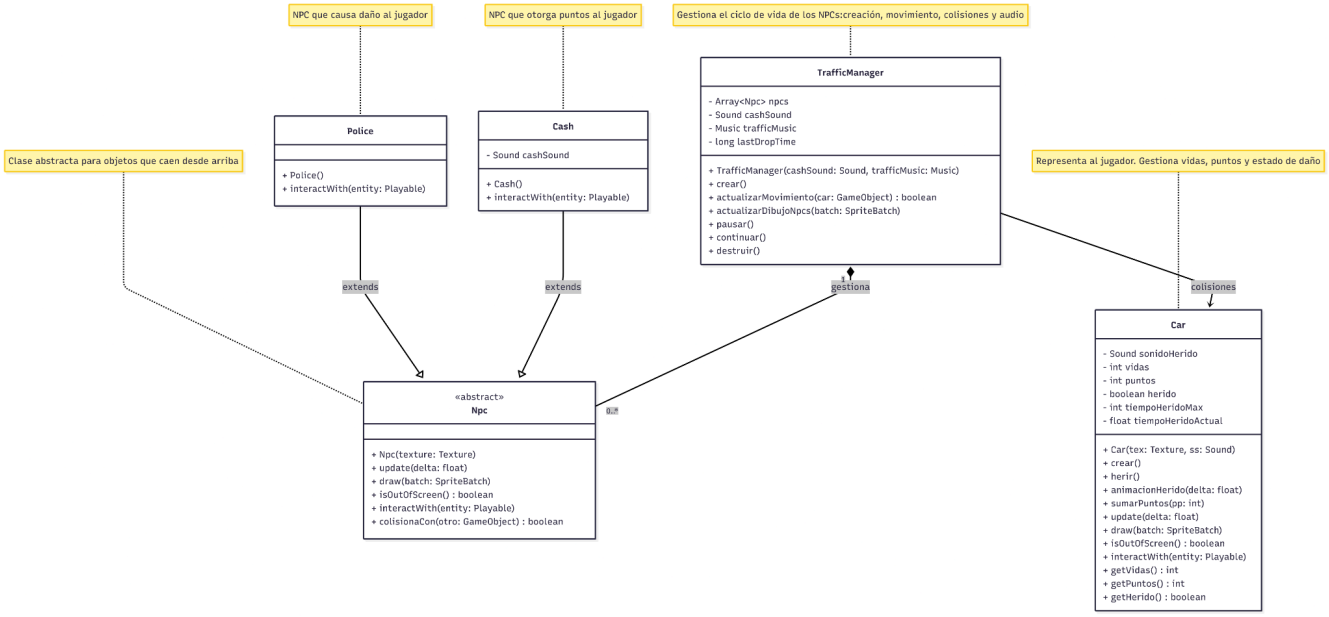
Juego base: Se basa en el patrón “catch/avoid” típico de los ejemplos de LibGDX, adaptado a una dinámica de desplazamiento horizontal del jugador y caída vertical de NPCs.

Arquitectura: Las entidades del mundo comparten estructura mediante la clase abstracta GameObject (posición, tamaño, textura y colisiones) y un contrato común mediante la interfaz Entity (dibujar, actualizar, interacción y verificación de límites). La clase abstracta Npc representa objetos que caen; Police y Cash son implementaciones concretas. Car representa al jugador e implementa las interfaces necesarias para integrarse con la lógica de juego y el HUD. TrafficManager gestiona el ciclo de vida de los NPCs (aparición, movimiento, colisiones y audio) y actúa como contexto para aplicar interacciones polimórficas. La navegación se resuelve con pantallas LibGDX (MainMenuScreen, GameScreen, PausaScreen y GameOverScreen) coordinadas por GameLluviaMenu.

Extensiones previstas: Añadir nuevos tipos de NPC anotados con @NPCType (por ejemplo, obstáculos adicionales), que serán detectados automáticamente por el sistema de registro. Balancear parámetros del juego ajustando valores en los archivos de configuración (schema/GameConfig, schema/CarConfig y schema/NpcConfig). Mejorar la experiencia de usuario con efectos visuales, y persistir el HighScore en almacenamiento local como mejora futura.

GM1.3 Diseño UML y referencia a código

El diagrama en uml/UML-Cars.png representa las principales clases y relaciones: GameObject como clase base, Npc como clase abstracta para elementos que caen, Car como entidad controlable por el jugador, y las implementaciones concretas Police y Cash. También se incluye la interfaz Entity, la interfaz Playable y las pantallas que gestionan el flujo del juego. Las clases se ubican en el paquete puppy.code y la configuración en puppy.code.schema.



GM1.4 Clase abstracta padre de al menos dos clases y uso en contexto

La clase abstracta GameObject es padre de Car y Npc. A su vez, Npc es padre de Police y Cash. Estas jerarquías se utilizan en contexto en TrafficManager, que recibe el jugador como GameObject (y también lo trata como Playable) para gestionar colisiones y aplicar efectos sin conocer el tipo concreto del NPC.

GM1.5 Interfaz implementada por al menos dos clases y uso en contexto

La interfaz Entity es implementada por Car y por todas las clases que extienden Npc. La interfaz Playable es implementada por Car y es utilizada por los NPCs a través del método de interacción para herir al jugador o sumar puntos. TrafficManager invoca la interacción polimórficamente, sin acoplarse a tipos concretos, usando la interfaz Entity como contexto.

GM1.6 Encapsulamiento y principios de orientación a objetos

Encapsulamiento: Los atributos relevantes se mantienen privados o protegidos y se acceden por métodos específicos; por ejemplo, las vidas, puntos y estado de “herido” del jugador se gestionan dentro de Car sin exponer campos públicos.

Abstracción: GameObject y Npc agrupan comportamiento y datos comunes, mientras que Entity define el comportamiento para participar del ciclo de actualización y renderizado, y para interactuar en colisiones.

Polimorfismo: Police y Cash especializan el comportamiento de interacción con el jugador, y TrafficManager ejecuta dicha interacción sin conocer el tipo de NPC concreto.

Responsabilidad única: TrafficManager se encarga exclusivamente de la gestión de NPCs, GameScreen orquesta el bucle de juego y la navegación entre pantallas de juego/pausa/game over, y Car se ocupa del estado y comportamiento del jugador.

GM1.7 Utilización de GitHub

Esto se logra en: https://github.com/Schiz0idCat/cars