**Mô tả dự án**

Trong trò chơi Zing Speed Mobile, người chơi sẽ được trải nghiệm cuộc đua tốc độ hồi hộp và kịch tính trên các địa hình đa dạng. Trong một thế giới đua xe điện tử, các chiếc xe đua không chỉ là phương tiện để vượt qua đường đua mà còn là biểu tượng của sức mạnh và tài năng của các tay đua.

Lớp Xe Đua trong trò chơi đóng vai trò quan trọng, với những thuộc tính đặc trưng như tốc độ, khả năng tăng tốc, độ bền và khả năng điều khiển trên mọi loại địa hình. Mỗi chiếc xe được định danh bằng mô hình và loại, từ "Ferrari" sang "Lamborghini", mang đến sự đa dạng và cá nhân hóa cho trải nghiệm đua xe.

Các Vật Phẩm trong trò chơi cung cấp thêm chiến lợi phẩm cho người chơi, từ Turbo để tăng tốc độ đột ngột, Nitro để tăng tốc độ tạm thời, đến Shield để bảo vệ xe khỏi sự va chạm. Mỗi vật phẩm mang lại hiệu ứng đặc biệt khi sử dụng, kéo dài trong một khoảng thời gian nhất định và có sẵn trong một số lượng giới hạn.

Trên các Địa Hình đa dạng như thành phố, sa mạc và rừng rậm, người chơi sẽ phải đối mặt với các thách thức khác nhau. Độ khó của mỗi địa hình sẽ ảnh hưởng đến tốc độ và khả năng điều khiển của xe đua, trong khi các chướng ngại vật trên đường đua sẽ thách thức kỹ năng lái xe của người chơi.

Trong vai trò của Người Chơi, người chơi sẽ tham gia vào các cuộc đua, cạnh tranh với những đối thủ khác để giành chiến thắng và vinh quang. Họ sẽ có cơ hội cải thiện lever của mình thông qua việc chiến thắng các cuộc đua và thu thập các vật phẩm trong trò chơi. Mỗi người chơi cũng sẽ có thể chọn cho mình một chiếc xe yêu thích và quản lý danh sách vật phẩm một cách thông minh để tối ưu hóa hiệu suất trong cuộc đua.

Trong tựa game này, mỗi trận đấu đều là một thử thách mới, và người chiến thắng sẽ là người có kỹ năng lái xe xuất sắc nhất, sử dụng chiếc xe phù hợp và tận dụng mọi cơ hội để vươn tới đích đầu tiên. Chính sự kết hợp giữa tốc độ, kỹ năng và chiến thuật sẽ quyết định ai là vị vua của đường đua trong trò chơi này.

Trong dự án xây dựng các lớp vật thể cho trò chơi Zing Speed Mobile, chúng ta sẽ tiến hành xây dựng các lớp ảo và các interface để định nghĩa các hành vi chung, sau đó xây dựng các lớp kế thừa từ các lớp ảo và implement các interface. Dưới đây là mô tả chi tiết về cách triển khai và các kỹ thuật được sử dụng:

**Xây dựng các lớp ảo:**

**1. Lớp ảo RacingObject:**

Chứa thuộc tính chung như mã, tên và mô tả cho các vật thể trong trò chơi.

Cung cấp các phương thức cơ bản để đọc và cập nhật thông tin của các vật thể.

**Xây dựng các interface:**

**1. Interface Effectable:**

Định nghĩa phương thức applyEffect() để áp dụng hiệu ứng của vật phẩm lên các đối tượng trong trò chơi.

**2. Interface Movable:**

Định nghĩa phương thức move() để di chuyển các đối tượng trên địa hình. Xây dựng các lớp kế thừa từ lớp ảo:

**Xây dựng các lớp kế thừa từ lớp ảo:**

**1. Lớp RacingCar:**

Kế thừa từ lớp RacingObject và implement interface Movable.

Định nghĩa các thuộc tính và phương thức đặc trưng cho xe đua như tốc độ, khả năng tăng tốc và điều khiển.

**2. Lớp Item:**

Kế thừa từ lớp RacingObject và implement interface Effectable.

Chứa các thông tin về vật phẩm và hiệu ứng của nó lên các đối tượng trong trò chơi.

**3. Lớp Terrain:**

Kế thừa từ lớp RacingObject

Định nghĩa các thuộc tính và phương thức đặc trưng cho địa hình như loại, độ khó và chiều dài.

**4. Lớp Player:**

Kế thừa từ lớp RacingObject.

Chứa thông tin về người chơi và quản lý xe đua và vật phẩm của họ.

Triển khai các kỹ thuật override, đa hình và quản lý danh sách các đối tượng:

Override các phương thức cần thiết từ các lớp ảo để tùy chỉnh hành vi của các đối tượng.

Sử dụng đa hình để xử lý các tương tác giữa các đối tượng trong trò chơi.

Quản lý danh sách các đối tượng bằng cách sử dụng các cấu trúc dữ liệu như ArrayList để thêm, xóa và cập nhật các đối tượng trong trò chơi.

**Thực hiện các thao tác CRUD:**

Tạo, đọc, cập nhật và xóa các đối tượng trong trò chơi thông qua các phương thức phù hợp của từng lớp.

Qua việc xây dựng các lớp và interface, triển khai các kỹ thuật override và đa hình, cũng như quản lý danh sách các đối tượng và thực hiện các thao tác CRUD, chúng ta có thể tạo ra một hệ thống vật thể linh hoạt và mạnh mẽ cho trò chơi Zing Speed Mobile.

**Lớp Xe Đua (RacingCar):**

speed: double - Tốc độ hiện tại của xe đua.

acceleration: double - Khả năng tăng tốc của xe.

durability: int - Độ bền của xe, tức là khả năng chịu va đập và hỏng hóc.

handling: double - Khả năng điều khiển của xe trên các loại địa hình khác nhau.

model: String - Mô hình hoặc loại của xe.

nitroCapacity: double - Dung lượng nitro, nếu có, cho phép tăng tốc độ tạm thời.

**Lớp Vật Phẩm (Item):**

name: String - Tên của vật phẩm.

effect: String - Hiệu ứng hoặc tác dụng của vật phẩm lên xe đua.

duration: int - Thời gian vật phẩm có hiệu lực sau khi sử dụng.

quantity: int - Số lượng của vật phẩm trong trò chơi.

**Lớp Địa Hình (Terrain):**

type: String - Loại địa hình.

difficulty: int - Độ khó của địa hình.

length: double - Độ dài của đường đua.

obstacles: ArrayList<String> - Danh sách các chướng ngại vật trên đường đua.

**Lớp Người Chơi (Player):**

name: String - Tên người chơi.

lever: int – Cấp độ của người chơi trong trò chơi.

currentCar: RacingCar - Xe đua hiện đang được người chơi sử dụng.

inventory: ArrayList<Item> - Danh sách các vật phẩm mà người chơi đang sở hữu.

**Lớp Trò Chơi (Game):**

players: ArrayList<Player> - Danh sách người chơi tham gia trò chơi.

currentTerrain: Terrain - Địa hình đang được chọn cho trận đấu hiện tại.

timeLimit: int - Thời gian giới hạn của mỗi trận đấu.

winner: Player - Người chơi chiến thắng trong mỗi trận đấu.

**Sơ đồ uml thể hiện bằng chữ**

**Class abstract RacingObject {**

- id: int

- name: String

- description: String

+ RacingObject(id: int, name: String, description: String)

+ getId(): int

+ getName(): String

+ getDescription(): String

+ setId(id: int): void

+ setName(name: String): void

+ setDescription(description: String): void

}

**interface Effectable {**

+ applyEffect(): void

}

**interface Moveable** {

+ accelerate(): void

+ brake(): void

+ steerLeft(): void

+ steerRight(): void

+ stop(): void

}

**class RacingCar {**

- speed: double

- acceleration: double

- durability: int

- handling: double

- model: String

- nitroCapacity: double

+ RacingCar(speed: double, acceleration: double, durability: int, handling: double, model: String, nitroCapacity: double)

+ getSpeed(): double

+ getAcceleration(): double

+ getDurability(): int

+ getHandling(): double

+ getModel(): String

+ getNitroCapacity(): double

+ setSpeed(speed: double): void

+ setAcceleration(acceleration: double): void

+ setDurability(durability: int): void

+ setHandling(handling: double): void

+ setModel(model: String): void

+ setNitroCapacity(nitroCapacity: double): void

+ Nhap(): void

+ Xuat(): void

}

**class Item {**

- name: String

- effect: String

- duration: int

- quantity: int

+ Item(name: String, effect: String, duration: int, quantity: int)

+ getName(): String

+ getEffect(): String

+ getDuration(): int

+ getQuantity(): int

+ setName(name: String): void

+ setEffect(effect: String): void

+ setDuration(duration: int): void

+ setQuantity(quantity: int): void

+ Nhap(): void

+ Xuat(): void

}

**class Terrain {**

- type: String

- difficulty: int

- length: double

- obstacles: ArrayList<String>

+ Terrain(type: String, difficulty: int, length: double, obstacles: ArrayList<String>)

+ getType(): String

+ getDifficulty(): int

+ getLength(): double

+ getObstacles(): ArrayList<String>

+ setType(type: String): void

+ setDifficulty(difficulty: int): void

+ setLength(length: double): void

+ setObstacles(obstacles: ArrayList<String>): void

+ Nhap(): void

+ Xuat(): void

}

**class Player {**

- name: String

- lever: int

- currentCar: RacingCar

- inventory: ArrayList<Item>

+ Player(name: String, Lever: int, currentCar: RacingCar, inventory: ArrayList<Item>)

+ getName(): String

+ getLever(): int

+ getCurrentCar(): RacingCar

+ getInventory(): ArrayList<Item>

+ setName(name: String): void

+ setLever(lever: int): void

+ setCurrentCar(currentCar: RacingCar): void

+ setInventory(inventory: ArrayList<Item>): void

+ Nhap(): void

+ Xuat(): void

}

**class Game {**

- players: ArrayList<Player>

- currentTerrain: Terrain

- timeLimit: int

- winner: Player

+ Game(players: ArrayList<Player>, currentTerrain: Terrain, timeLimit: int, winner: Player)

+ getPlayers(): ArrayList<Player>

+ getCurrentTerrain(): Terrain

+ getTimeLimit(): int

+ getWinner(): Player

+ setPlayers(players: ArrayList<Player>): void

+ setCurrentTerrain(currentTerrain: Terrain): void

+ setTimeLimit(timeLimit: int): void

+ setWinner(winner: Player): void

+ Nhap(): void

+ Xuat(): void

}

Item --|> RacingObject

RacingCar --|> RacingObject

Terrain --|> RacingObject

Player --|> RacingObject

RacingCar --> Movable

Item --> Effectable

**Các câu bài tập**

Hãy viết một phương thức trong lớp RacingCar để tính toán và trả về tốc độ tối đa mà xe có thể đạt được, dựa trên tốc độ hiện tại và dung lượng nitro.

Xây dựng một phương thức trong lớp Item để áp dụng hiệu ứng của vật phẩm lên một đối tượng được chỉ định, với quản lý thời gian hiệu lực của vật phẩm.

Viết một phương thức trong lớp Terrain để tính toán và trả về tổng số chướng ngại vật trên đường đua, dựa trên danh sách các chướng ngại vật.

Hãy thực hiện một phương thức trong lớp Player để thêm một vật phẩm mới vào danh sách vật phẩm của người chơi và cập nhật số lượng tương ứng.

Xây dựng một phương thức trong lớp Game để kiểm tra và xác định người chơi chiến thắng dựa trên điểm số của họ và thời gian giới hạn.

Hãy triển khai một phương thức trong lớp RacingCar để xử lý việc tăng tốc khi người chơi sử dụng nitro, với việc giảm dung lượng nitro tương ứng.

Hãy thực hiện một phương thức trong lớp Game để quản lý thời gian giới hạn của mỗi trận đấu và thông báo khi thời gian đã hết.

Làm thế nào bạn sẽ sử dụng các cấu trúc dữ liệu như ArrayList để quản lý danh sách các đối tượng trong trò chơi và thực hiện các thao tác CRUD (Tạo, Đọc, Cập nhật, Xóa) một cách hiệu quả?