Mille Bornes

The Racing Card Game: Project 1 Write-up

Adrian Carballo

CSC-17C

5 / 7 / 2025

Introduction

I am coding the classic Mille Bornes Racing Card Game. I chose this game because on top of being very entertaining, it was a unique challenge to code this type of card game using different cards that have relationships with each other. I spent about 4 weeks in total cementing the rules, drawing up concepts, and coding the project. The program has around 510 lines of code, features two classes, and utilizes the STL which includes the list, random, algorithm, stack, and queue libraries.

The project is located on GitHub:

https://github.com/Acey-Softworks/MilleBornes

Approach to Development

The first thing necessary was to cement the rules into my head. Since there are four different types of cards and piles, the best route to cementing the rules will be by making a flowchart. Making a flowchart showing the possible moves for each type of card and pile made the development process easier as the logic will already be complete when it is time to code.

The program was completed in 6 versions. In the 1st version, I made the deck of cards as a list and used a structure to hold card information. In the 2nd version, I tweaked the deck initialization, and created function prototypes for shuffling the deck. I created a shuffle algorithm in the 3rd version which was redone to include the random library from the STL in the 4th version. The 4th version also introduced a Player class to hold player information and player hands and I changed the Card structure to a Card class. A gameplay loop was added in the 5th version for two players which was fixed in the final version. The final version completes the program by completing the validation of the cards played for each type of card and deck.

Game Rules

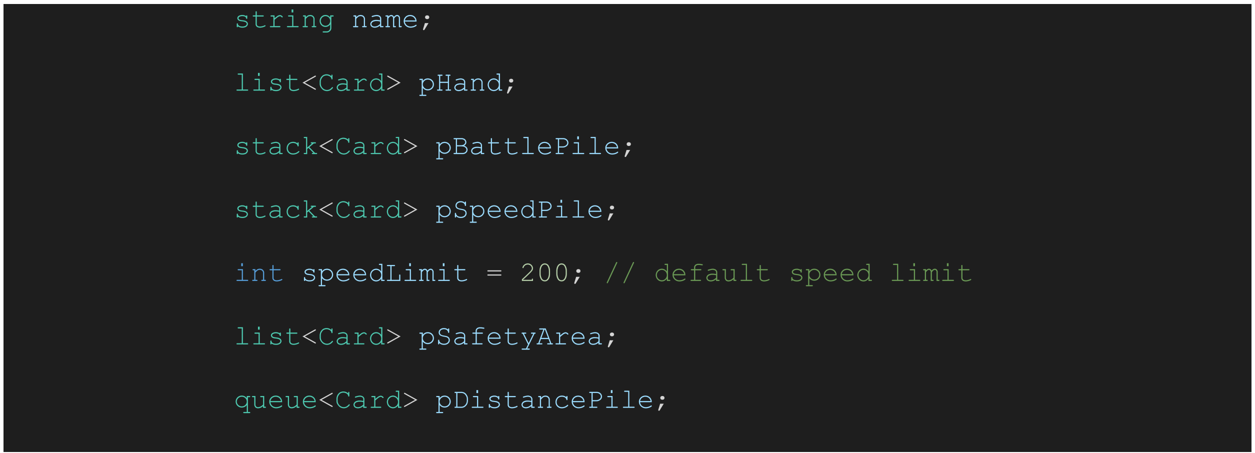
Mille Bornes is a classic racing card game released in France in 1954. Because of its age, there are plenty of different versions of the game available. I created the game based on the 2 player version from Winning Moves. There are 4 card types in the deck. Hazard cards stop opponents from moving forward until they are remedied by a Remedy card. Safety cards act as preventions for Hazard cards. For example, the "Flat Tire" hazard card can be remedied by the "Spare Tire" remedy card or which could have been prevented by the "Puncture Proof" safety card. Finally, Distance cards allow the player to score miles to eventually achieve exactly 700 miles (for 2 players) and win the game. The game also features scoring based on each card played which was excluded from my program. Only reaching 700 miles will the player win the game.

There are also 4 areas where the cards played by the player must be displayed, that is the Safety Area, Speed Pile, Battle Pile, and Distance Pile. The Safety Area displays the safety cards played. The Speed Pile displays "Speed Limit" cards and "End Of Limit" remedy cards. The battle pile displays hazard and remedy cards. The Distance Pile displays the mile cards.

A typical turn of Mille Bornes consists of drawing one card from the deck, playing a card that is valid to the table or discarding a card. The player always has 6 cards at the end of their turn. There are some rules to playing cards based on the type. A "Roll" card must be played onto the battle pile in order to start playing Distance cards. When a hazard card is played onto an opponent's battle pile, the receiving player must remedy the card and then play the “Roll” card to continue playing miles. A “Roll” card isn’t needed if the player has the “Right of Way” safety card in their safety area. Speed limit cards are played on the opponents speed pile which can be remedied by an “End of Limit” card.

Description of Code

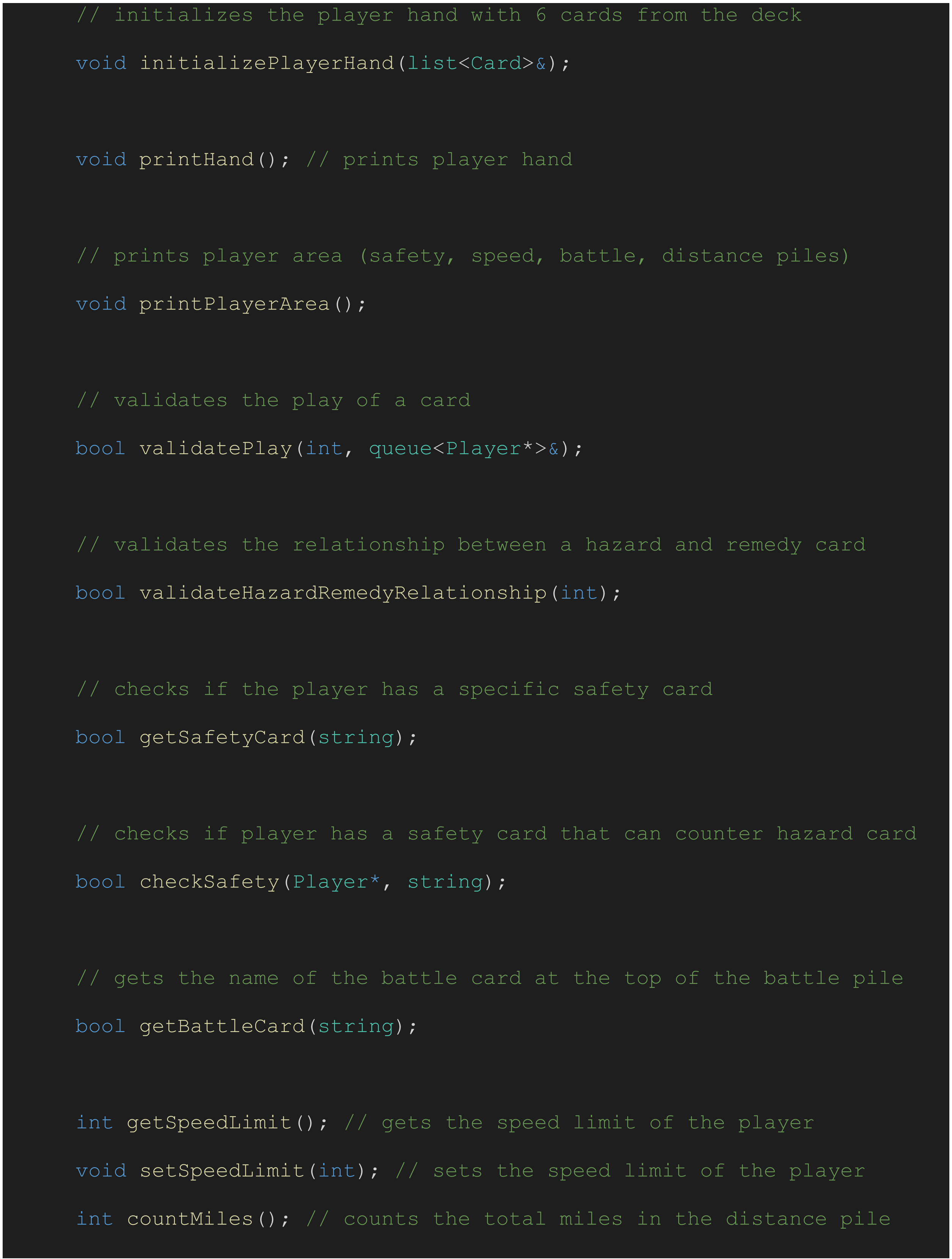
The project is organized into 4 files: main.cpp, Card.hpp, Player.hpp, and Player.cpp. The Card class only holds the card information, which includes the name and value of the card. The Player class holds the player information and player decks:



```
string name;

list<Card> pHand;

stack<Card> pBattlePile;

stack<Card> pSpeedPile;

int speedLimit = 200; // default speed limit

list<Card> pSafetyArea;

queue<Card> pDistancePile;
```

Player.hpp - Member variables

Different decks serve different purposes, therefore, their containers will be different. The Player Hand and Safety Area can be implemented using a list, for easy traversing. The Battle Pile and Speed Pile can be implemented using a stack, since only the top card of those piles are relevant. Using a queue for the Distance pile is convenient for adding the mileage together.



```
    // initializes the player hand with 6 cards from the deck
    void initializePlayerHand(list<Card>&);

    void printHand(); // prints player hand

    // prints player area (safety, speed, battle, distance piles)
    void printPlayerArea();

    // validates the play of a card
    bool validatePlay(int, queue<Player*>&);

    // validates the relationship between a hazard and remedy card
    bool validateHazardRemedyRelationship(int);

    // checks if the player has a specific safety card
    bool getSafetyCard(string);

    // checks if player has a safety card that can counter hazard card
    bool checkSafety(Player*, string);

    // gets the name of the battle card at the top of the battle pile
    bool getBattleCard(string);

    int getSpeedLimit(); // gets the speed limit of the player
    void setSpeedLimit(int); // sets the speed limit of the player
    int countMiles(); // counts the total miles in the distance pile
```

Player.hpp - Member function prototypes

Sample Input/Output

```
Player 1's turn!
Player 1 draws a card from the deck...

Player 1's Area:
Safety Area:
SAFETY: EXTRA TANK

Speed Pile: Empty

Battle Pile:
REMEDY: ROLL!

Distance Pile:
200mi, 200 MILES TOTAL.

Hand:
1. DISTANCE: 25mi
2. HAZARD: STOP!
3. DISTANCE: 25mi
4. DISTANCE: 100mi
5. DISTANCE: 100mi
6. REMEDY: GAS
7. HAZARD: SPEED LIMIT 50mph
0. Discard a card
```

```
Please select a card to play from your hand (0 - 7): 7


Opponents speed limit set to: 50mph

Card played.
```

The output shows the player area and hand. The only input necessary is playing a card number (highlighted in green). “Invalid move” will be printed if the move is found to be invalid.

Checkoff Sheet

1. Container Classes
    a. List

    Lists were used for the deck of cards, the players hand, and the players safety area (main.cpp; line 46, Player.hpp; lines 17 & 20)

    b. Stack

    Stacks were used for the players battle and speed piles (Player.hpp; lines 17 & 18)

    c. Queue

    Queues were used for the player gameplay loop (main.cpp; line 65) and the players distance pile (Player.hpp; line 21)

    d. Associative containers (set, map, hash) were not used in the program.

2. Iterators

   a. Forward iterator

Forward iterators were used to iterate through lists, like the player hand and safety area. Using the advance() function, we are able to iterate forward through the lists, since lists do not contain indices. The advance() function is used in the validatePlay() function (Player.cpp; line 3) and validateHazardRemedyRelationship() function (Player.cpp; line 153).

   b. Random access iterator

The shuffle() function requires a random access iterator. The shuffle function is used to shuffle the deck in the shuffleDeck() function (main.cpp; line 158).

3. Algorithms

   a. for_each

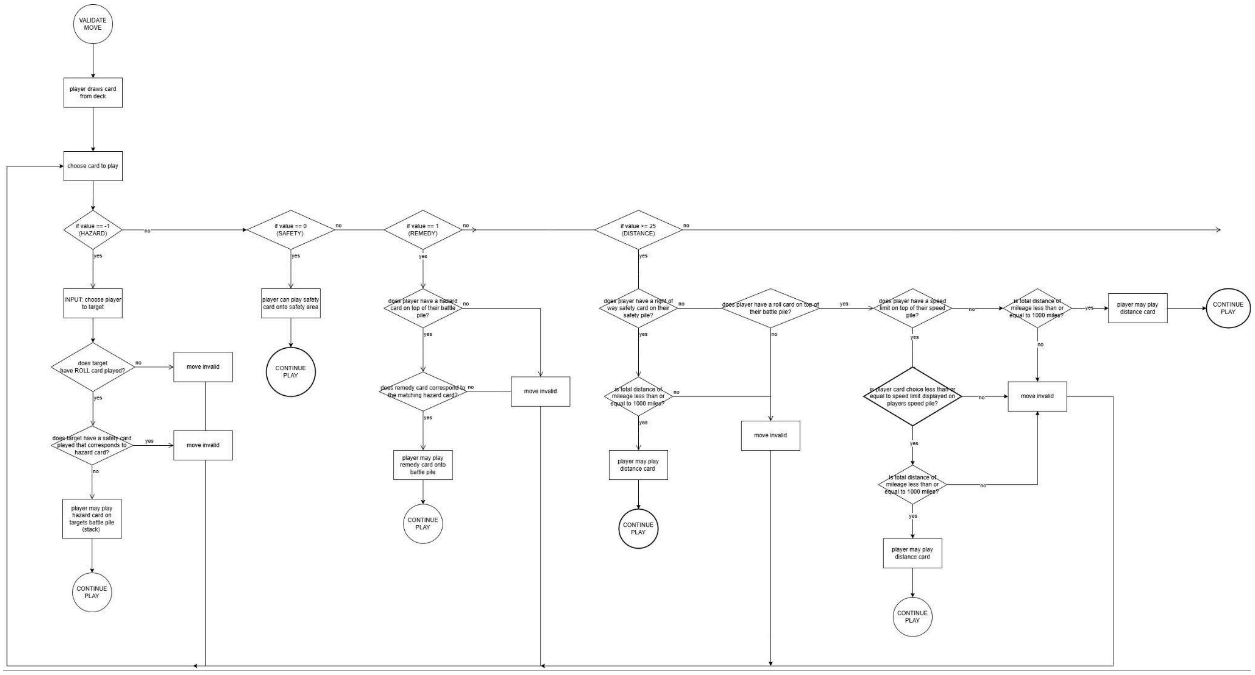
The for_each loop is a range based for loop as seen in the getSafetyCard() function (Player.cpp; line 178). The loop traverses the list and looks at each element in the list.

   b. random_shuffle

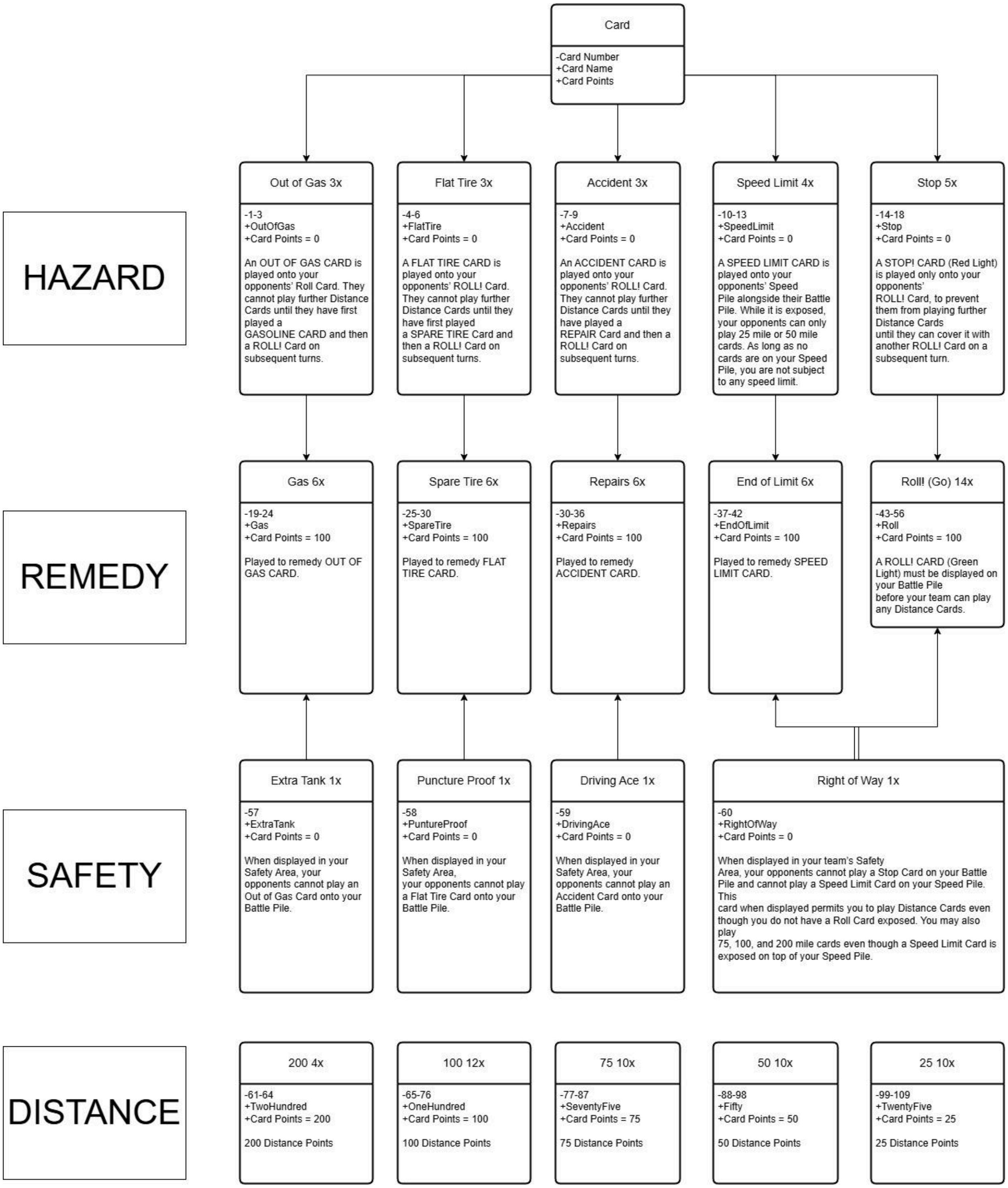
The random_shuffle function has since been deprecated and removed from C++ and has been replaced with the shuffle function. The shuffle function is used to shuffle the deck in the shuffleDeck() function (main.cpp; line 158).

   c. Organization algorithms were not used in the program.

Documentation of Code


VALIDATE MOVE
player draws card from deck
choose card to play
if value == -1 (HAZARD)
no
yes
INPUT: choose player to target
does target have ROLL card played?
no
move invalid
yes
does target have a safety card played that corresponds to hazard card?
yes
move invalid
no
player may play hazard card on targets battle pile (stack)
CONTINUE PLAY
if value == 0 (SAFETY)
no
yes
player can play safety card onto safety area
CONTINUE PLAY
if value == 1 (REMEDY)
no
yes
does player have a hazard card on top of their battle pile?
no
yes
does remedy card correspond to the matching hazard card?
no
move invalid
yes
player may play remedy card onto battle pile
CONTINUE PLAY
if value >= 25 (DISTANCE)
no
yes
does player have a right of way safety card on their safety pile?
no
yes
is total distance of mileage less than or equal to 1000 miles?
no
yes
player may play distance card
CONTINUE PLAY
does player have a roll card on top of their battle pile?
yes
no
move invalid
does player have a speed limit on top of their speed pile?
no
yes
is player card choice less than or equal to speed limit displayed on players speed pile?
no
yes
is total distance of mileage less than or equal to 1000 miles?
no
yes
player may play distance card
CONTINUE PLAY
is total distance of mileage less than or equal to 1000 miles?
yes
no
player may play distance card
CONTINUE PLAY
move invalid

Flowchart 1: Logic to Validate Move



Card

-Card Number
+Card Name
+Card Points

HAZARD

Out of Gas 3x

-1-3
+OutOfGas
+Card Points = 0

An OUT OF GAS CARD is played onto your opponents' Roll Card. They cannot play further Distance Cards until they have first played a GASOLINE CARD and then a ROLL! Card on subsequent turns.

Flat Tire 3x

-4-6
+FlatTire
+Card Points = 0

A FLAT TIRE CARD is played onto your opponents' ROLL! Card. They cannot play further Distance Cards until they have first played a SPARE TIRE Card and then a ROLL! Card on subsequent turns.

Accident 3x

-7-9
+Accident
+Card Points = 0

An ACCIDENT CARD is played onto your opponents' ROLL! Card. They cannot play further Distance Cards until they have played a REPAIR Card and then a ROLL! Card on subsequent turns.

Speed Limit 4x

-10-13
+SpeedLimit
+Card Points = 0

A SPEED LIMIT CARD is played onto your opponents' Speed Pile alongside their Battle Pile. While it is exposed, your opponents can only play 25 mile or 50 mile cards. As long as no cards are on your Speed Pile, you are not subject to any speed limit.

Stop 5x

-14-18
+Stop
+Card Points = 0

A STOP! CARD (Red Light) is played only onto your opponents' ROLL! Card, to prevent them from playing further Distance Cards until they can cover it with another ROLL! Card on a subsequent turn.

REMEDY

Gas 6x

-19-24
+Gas
+Card Points = 100

Played to remedy OUT OF GAS CARD.

Spare Tire 6x

-25-30
+SpareTire
+Card Points = 100

Played to remedy FLAT TIRE CARD.

Repairs 6x

-30-36
+Repairs
+Card Points = 100

Played to remedy ACCIDENT CARD.

End of Limit 6x

-37-42
+EndOfLimit
+Card Points = 100

Played to remedy SPEED LIMIT CARD.

Roll! (Go) 14x

-43-56
+Roll
+Card Points = 100

A ROLL! CARD (Green Light) must be displayed on your Battle Pile before your team can play any Distance Cards.

SAFETY

Extra Tank 1x

-57
+ExtraTank
+Card Points = 0

When displayed in your Safety Area, your opponents cannot play an Out of Gas Card onto your Battle Pile.

Puncture Proof 1x

-58
+PuntureProof
+Card Points = 0

When displayed in your Safety Area, your opponents cannot play a Flat Tire Card onto your Battle Pile.

Driving Ace 1x

-59
+DrivingAce
+Card Points = 0

When displayed in your Safety Area, your opponents cannot play an Accident Card onto your Battle Pile.

Right of Way 1x

-60
+RightOfWay
+Card Points = 0

When displayed in your team's Safety Area, your opponents cannot play a Stop Card on your Battle Pile and cannot play a Speed Limit Card on your Speed Pile. This card when displayed permits you to play Distance Cards even though you do not have a Roll Card exposed. You may also play 75, 100, and 200 mile cards even though a Speed Limit Card is exposed on top of your Speed Pile.

DISTANCE

200 4x

-61-64
+TwoHundred
+Card Points = 200

200 Distance Points

100 12x

-65-76
+OneHundred
+Card Points = 100

100 Distance Points

75 10x

-77-87
+SeventyFive
+Card Points = 75

75 Distance Points

50 10x

-88-98
+Fifty
+Card Points = 50

50 Distance Points

25 10x

-99-109
+TwentyFive
+Card Points = 25

25 Distance Points

Flowchart 2: First Draft of Card Relationships