Assignment 2

Flamme Rouge

This project will allow you to implement a computer game, with some simple graphics. It is a bicycle race game, where you will implement some simple artificial intelligence for a computer player.

This project is worth 7.5% of your final grade. You must do this project on your own. The submission deadline is the last teaching day of semester, Friday 13th October 2023 at 5pm. Your submission will be marked out of 12 in the lab, then your C# code will be marked out of 8 after you submit your program.

Project Specification:

You have been asked to write a computer version of the "Flamme Rouge" game, by Asger Harding Granerud. From the game description: "In the cycling world, a red flag—Flamme Rouge—signals the last kilometer in a race. It is a sign of the last stretch, and the competing cyclists know to give their all! Gathered on the outskirts of Paris, cyclists from all around the world have arrived to participate in a great race. Everyone wants to ride underneath that red flag towards fame and glory! Spectators are bound to witness a true bout of stamina and strategy. May the best team win!"

Your program should implement a simplified 2-player version of the game where the user plays against the computer, or another player. In our simplified version, the track will be straight, and shorter, with the cyclists having correspondingly fewer cards.

Overview[1]

Flamme Rouge is a fast-paced, tactical bicycle racing game where each player controls a team of two riders: a *Rouleur* and a *Sprinteur*. The players' goal is to be the first to cross the finish line with one of their riders. If more than one rider crosses the line in the same round, the one who gets furthest across wins. Players move their riders forward by drawing and playing numbered cards, which show how far each rider moves.

[1] Rules modified from https://boardgamegeek.com/boardgame/199478/flamme-rouge/

Preparation for a game

Shuffle a separate deck for each players *Sprinteur* and *Rouleur* rider. Generate a track of appropriate length (with mountains if they are being used). Randomly place the players cyclists in the squares before the start line.

Playing a round

The race is played over several rounds. Each round has three phases:

- In the Energy Phase players (simultaneously) draw 4 and play 1 card face down for one of their riders first, then the other.
- In the Movement Phase all played cards are revealed and riders move according to their card, starting with the frontmost rider.
- In the End Phase the played cards are removed, unplayed cards are recycled, then slipstreaming and exhaustion is applied.

If a riders energy deck runs out of cards, shuffle the recycled cards to form a new deck. Keep drawing until you have four cards in hand.

Read the rulebook for more details on the movement, slipstreaming and exhaustion rules.

Game Cards and Track Length

Flamme Rouge has two decks of cards – one for each type of rider. The *Sprinteur* deck has cards with values 2, 3, 4, 5, and 9. The *Rouleur* deck has cards of values 3, 4, 5, 6, and 7. There are also *Exhaustion* cards with a value 2. The original game has 3 of each card in each deck, and the track length is 68 squares in length (plus 4 squares before the start and 4 squares after the finish).

For a short game, I suggest each deck has only 1 of each card, and the track is 22 squares from start to finish. For a middle length game, I suggest 2 of each card and a track length of 45.

Mountains

Mountains change how the track works: Ascents (red edges) make 5 the highest movement, and have no slipstreaming. Descents (blue edges) make 5 the minimum movement. Mountains normally are a block of 2-5 squares which are either Ascents or Descents, which sometimes are together on the track, but not necessarily. The Start and Finish squares are always on normal flat track.

Read the rulebook for more details on how Ascents stop movement if your rider passes onto an Ascent square with a movement higher than 5.

Robot player

The game can be written where humans control both players, or you can write a robot player to play against.

At each turn the robot player needs to choose a card for each rider. A purely random robot player would not be that interesting to play against. So, your robot player must try to have some artificial intelligence, making its choice based on the riders' distance from the frontmost rider or the finish line, the track type (if mountains are used) and possible slipstreaming from its other rider. You may want to let the user choose from robots that play in different styles.

Tasks

Write a program that satisfies the specifications described above. It should make use of all of the programming features that you have learnt so far, including using classes and methods to structure your program and XML documentation where appropriate.

The project requires you to both get it verified in the lab and to hand in the source code. Half of the grade will be based on how the program runs and half on the object design, structure, code style and documentation.

Suggested Steps

We suggest that you build a simplified version of the game first, and then add in more features as time permits. Concentrate on getting the objects and simple turn play right before adding fancy graphics and controls.

Start with only one (type of) rider, and flat track. This means that the deck is simplified and you can concentrate on movement and game play.

You may want to generate the cards randomly to start with, and then add code that keeps track of cards to get the correct ratio of cards. It is easier to generate the cards in order and shuffle them than to randomly generate a card and see if it is one of the unused cards.

For testing purposes, specify how many cards (and track length) are used in the game, so a game can be shorter.

It is often easier to control both players, adding a computer player once you have a basic game going. Once you have a stupid computer player, then add more intelligence and heuristics as you have time.

COMPX102-23B

Assignment 2 Hand-in due 5pm 13th October 2023

Compress (Zip) the Visual Studio folder with your program code and submit it via Moodle.

Student Declaration of Originality

I declare that the program which I have had verified and submitted in Moodle is entirely my own work. I have not worked together with any other people. I have suitably acknowledged (referenced) any parts of other programs that I have used. I understand that if I have breached the above conditions, I will be sent to the University Disciplinary Committee.

Name: _____

ID Number: _____

Signed: _____

Date: _____

Functionality and Usability (demonstrated in the lab) _____ /12 marks

	Minimal or none	Partial, simple or glitchy	Works
Deals hands with random (order of) cards	0	½	1
Player can play a card for each rider	0	½	1
Riders moved to new positions (in race order)	0	½	1
Resolves blocked move correctly, also keep right	0	½	1
Recycled cards are shuffled into new deck	0	½	1
Exhaustion cards awarded	0	½	1
Slipstreaming implemented	0	½	1
Mountains (ascents and descents) implemented	0	½	1
Game continues turns until race won	0	½	1
Second player or robot player implemented	0	½	1

	Missing or poor	Hard to use or understand	Easy to use
Usability: layout, choice of controls, feedback, etc	0	1	2

Bonus mark: Save & load game state _____ /+1 mark
Bonus mark: Supports > 2 players (or robot players) in game _____ /+1 mark

Coding Style (marked by tutor after zip submitted) _____ /8 marks

	Poor	Average	Good	Excellent
Object design, inheritance, composition, scope	0	1	2	3
Code style, user methods, code structure	0	1	2	3
Documentation: XML and inline comments	0	1	2	2

Total: _____ **/20 marks**