1/19/2020

Dangerous Racers

Video Game Atchitecture & Optimisation Assignment 1

Teodor Grigor – Student ID 19412013
UNIVERSITY OF NORTHAMPTON

Contents

1. About Game

Dangerous Racers is a racing game that can be played on computers. The player, after selecting a car and a track, can try to beat other cars (computer-controlled). Dangerous Racers was a very popular game at the time when it was launched; after several years appeared other games that look like this game or have the same mechanics.

This game has many levels and many playable cars; while you are on a race if you enter into another car then you will take an amount of damage from the car that you were involved in a crash with. If the 'health' of that car reaches 0, the car will explode and remain on track as an object – if you enter into the crashed cars you will collide with them.

The original game also had some pick-up objects dropped on the race track. If you take one of these items while you are racing you get the power of the item that you picked (nitro gets you extra speed, health increase your health). Also, while you are on a race you can not exit the race track – if you attempt to exit the race track you will collide into the objects that are near the track and your speed will decrease. I added some extra power-ups such as the explosion and reverse.

In my version of the game, there are only 3 levels – this means 3 racetracks and 3 playable cars. Also, my version of the game is a little bit different than the original because you can exit the track without any problems (you won't collide into the racetrack margins) but your speed will be decreased. Also, there are some checkpoints added on the racetrack and if you are driving anywhere but not on the racetrack your current lap will be marked as skipped.

2. Game Objectives

In this game the main objective is to win the race; the race is marked as won only if you manage to end up in the first place. If you are ending the race in another place than the first you will be able to retry that race as many times as you want. Therefore, the game objective is to race against AI cars (the computer-controlled cars) and prove that you are capable of winning in the first place.

Another objective is to unlock new levels because by unlocking different levels beside that they have a different race track you will capable of choosing from different cars as the playable car. Different cars have different health and speed; by passing a new level the cars will have great power and bigger health. For instance, the first car has a total of 200 health, while the second car has 250 and the third car has 300 points for the health. Therefore, by unlocking a new level you will unlock new cars with an addition of 50 points for the health than the car from the previous level.

3. Game Mechanics

Dangerous Racers can be played by only one player. In my version, the player can choose from a range of 3 playable cars (a red one, black one and yellow one) – each car has different health. In every race, the automated cars (the computer-based ones) are driving automatically by going to the next checkpoint from the track. The checkpoints are added programmatically for each level because whenever you attempt to race at a level it will have the same racetrack.

The collision is detected whenever a sprite collides with the other one. To detect the collision in the game is used the Pixel Collision Detection method. The collision is most used between the cars but is also used to know if the player cars collide with the objects that are near the racetrack or if the car exits the racetrack. Whenever the user tries to exit the racetrack the algorithm detects the collision between the player car and the grass and therefore we will decrease the maximum car speed to default speed.

In the game, we are using random power-ups that help the user or even sabotage him/her. The Power-ups are added randomly in the checkpoints so they can be picked by the player or even by the AI cars. For the AI cars, only the health pack and bomb power-ups work, while against the player all the four power-ups affect. The health pack increases the life (health) for the car that picked it up by 25, while the bomb (also called explosion) decrease the health for the car by 50. The third power-up that is available only to the player increases the speed one time only by 2; the reverse power-up change the keys that the user controls the car: for instance, if the player wants to go to the left then he/she needs to press the right key and for the right is the opposite.

4. Screen flow diagrams, Characters, NPC, Background design

In the game are only 3 playable cars (each one has their information such as health and speed) and another car as the object that is near the track, the ambulance.

Fig 1 Black Car

Fig 2 Red Car

Fig 3 Ambulance

Fig 4 Orange Car

In the game are 3 racetracks and each one has his mask.

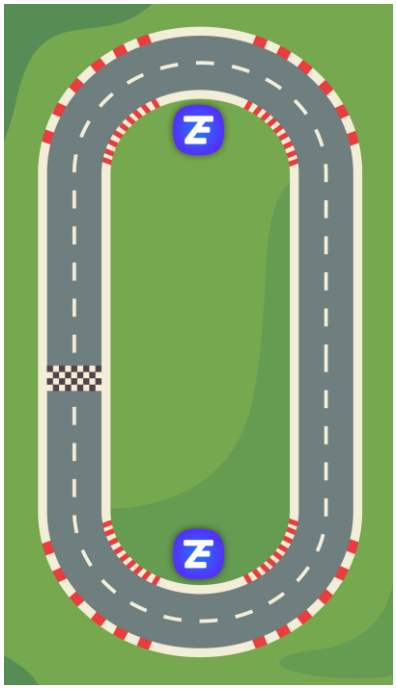

Fig 5 Race Track Lvl 1

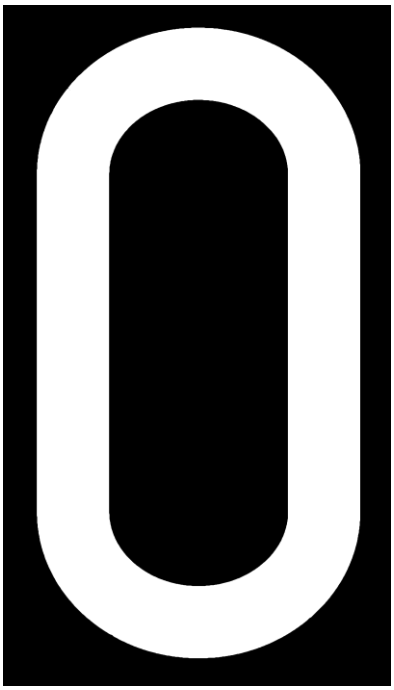

Fig 6 Mask Track Lvl 1

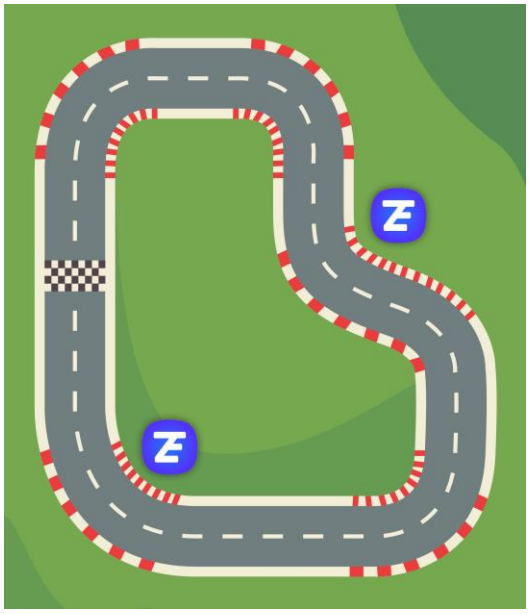

Fig 7 Race Track Lvl 2

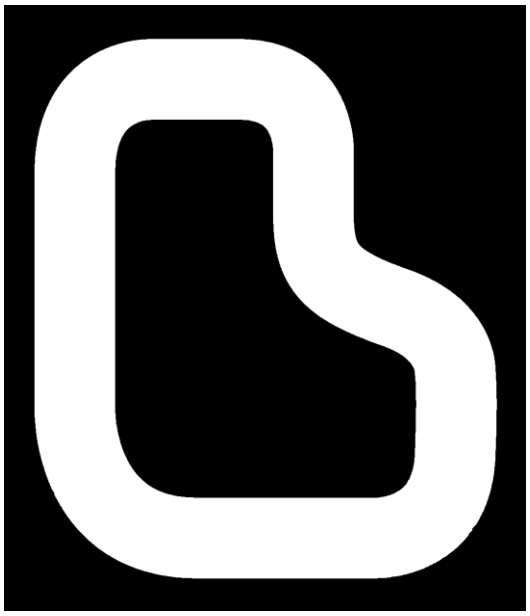

Fig 8 Mask Track Lvl 2

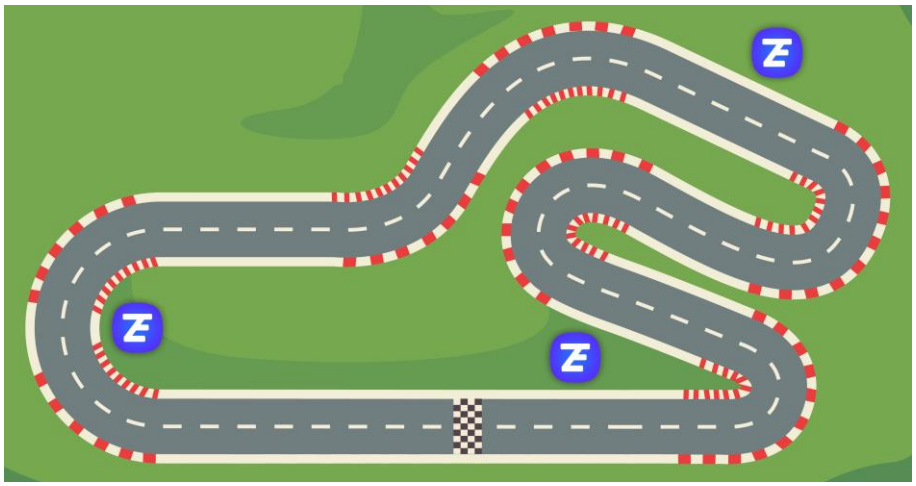

Fig 9 Race Track Lvl 3

Fig 10 Mask Track Lvl 3

5. Pseudo design

Run the game

Game Screen Selector:
If
    the game started
Then
    will be shown the splash screen for 2 seconds
    after 2 seconds we show the game menu
Else if
    the current screen is the game menu
Then
    will be shown the game menu until a button is clicked

Splash Screen:
Show the splash screen
In the middle of the SplashScreen is the ZeoFlow Logo
After 2 seconds show the game menu screen

Game Menu Screen:
Show the game menu
Set an image as the background
Add the animated playable characters to the background
Draw the buttons
    Draw the button for selecting the level
    Draw the button for choosing the character
    Draw the button for learning how to play
In case that the button for selecting the level is pressed
    We show the select level screen
In case that the button for selecting the character is pressed
    We show the select car screen
In case that the button for selecting the level is pressed
    We show the how to play screen

How to Play Screen:
Show the how to play screen
At the top create a button for going back
Get as input the text that contains the information about how to play
Write on the screen the information
Draw the going back button
In case that the 'going back' button is pressed
    Then we show the game menu screen

Choose Car Screen:
Show the screen that contains all the playable cars
At the top create a button for going back
Draw all the playable cars

In case a car is clicked
    Then we select that car as the primary
Draw the going back button
In case that the ‘going back’ button is pressed
    Then we show the game menu screen

Select Level Screen:
Show the screen for selecting the level
At the top create a button for going back
Draw the going back button
In case that the ‘going back’ button is pressed
    Then we show the game menu screen
Draw all the levels for the game
Enable only the levels that were unlocked
In case that an unlocked level was clicked
    Then we show the game screen for the relevant level

Game Screen:
Show the game screen
Draw the background
Draw the game board that holds the balls
Draw the character at the bottom of the game board
If the left arrow is pressed
    Then turn the car to the left
Else if the right arrow is pressed
    Then turn the car to the right
If the top arrow is pressed
    Then increase the car speed
Else if down arrow is pressed
    Then decrease the car speed
If all the cars are destroyed except the one of the player
    Then the race is marked as won
If the race ends and our place is the first
    Then we pass the level
Else
    We lose the level
Draw the minimap

6. Implementation of game

The game contains:

- a background / graphical character
- the character moves by keyboard entry
- NPC are included
- the main character interacts with other NPC
- scoring system
- full interaction with NPC
- a classic linear game architecture
- all NPC move
- victory status – you need to finish on the first place
- a total of 3 levels
- 3 playable characters
- mini-map
- 4 power-ups

7. Testing

ID	Objective	Action (Test Case)	Test Steps	Expected Result	Actual Result	Pass/Fail	Pass/Fail	Pass/Fail	Pass/Fail	Pass/Fail	Severity	% Pass	Comments
1	Check Game Menu Buttons	Mouse Click	Click on the mouse in 5 different positions on the screen	Mouse clicked detected only in some particular areas	Even if it works in some particluar areas it is detected even if your are having the sfml window in background	Pass	Pass	Fail	Fail	Pass	Major	60%	The mouse should be detected only when the game window is the current window
2	Image in the center of the screen	Run game in splash screen	Resize the SFML Window (new width & height) 5 times while the game runs	The image should appear in th middle of the screen vertically and horizontally too	The image appear where it should be	Pass	Pass	Pass	Pass	Pass	N/A	100%	The method that centralise the image works properly
3	Load the assets using a method	Run the game	Load 5 different types of assets such as png, jpg, etc.	The assets should loaded properly	The assets were loaded properly	Pass	Pass	Pass	Pass	Pass	N/A	100%	This method should be added into a header so it could be easily imported in other projects
4	Check car left/right turn	Run game in game screen	Press the left and right keys while driving	Car should turn either right either left	Car turns in the direction pointed by the user, but if the speed is 0 it doesn't move	Pass	Pass	Fail	Pass	Fail	Low	60%	Overall the turn method works, but if the speed is null it is logical that the car shouldn't be able to turn
5	Check car can accelerate/deaccelerate	Run game in game screen	Press the top and down arrow keys	The cars speed should increase or decrease	Depending on the key pressed the car speed increases or decreases	Pass	Pass	Pass	Pass	Pass	N/A	100%	The car speed increases only to the max speed and decreases only to the max speed/2
6	Check speed car when it is out of the track	Run game in game screen	Try to drive at different speed outside the track	The car speed should decrease	The car speed decreases when driving forward, but when driving backwards the cars speed decreases constantly	Pass	Fail	Fail	Fail	Pass	Major	40%	When driving backwards outside of the track the car speed should decrease to 0

8. Algorithm Optimisation Strategy

In the creation of the game, I have not used other classes even if I could. I created different structures for every object that I used; there is a total of 5 structures, therefore I have used 5 different objects. The first structure is called 'TrackObjects' and it stores all the information necessary about the objects that are going to be drawn on the racetrack. The information required is the X and Y values – for the position, the angle – help us at pointing the direction and the sprite – it stores the image of the object. Another structure used, 'TracksBackground', it stores the sprite (image) for each racetrack. The 'CarModels' structure is used for storing the sprites for each playable car, while the 'PowerUps' structure is used for the Power-Ups object that is going to appear on the racetrack. This means that we will need to store their position and their type because they can be of 4 types. Not in the last, the 'Car' structure that is one of the most important parts of the entire code. It stores all the details about the cars that appear on the racetrack such as AI cars (computer-controlled cars) and the player car. Here we are storing the car position, speed, angle, the car id (which is a unique number of the car for identification), the currentCheckPoint which helps us to make sure that the player doesn't cheat or to know where the AI cars should go next, the lap, health an max health, the car sprite and the sprite for the crashed car. Inside the 'Car' structure we have a lot of methods that are helping us to manipulate the car variables in different cases. The first method is the restart one that initializes the car struct (this is used before starting a new race). Its role is to set the speed, angle, and lap to 0. Another method is the one that initializes the car values such as the car sprite, the sprite for the crashed car, and the health. Another method is the one that helps us at calculating the car place. Inside the 'Car' structure we have methods that are increasing or decreasing the car speed based on the scenario. In the last, we have one of the most important methods, the one that checks the checkpoint and the one that makes the car move.

For using the power-ups properly in the game in the code can be found three methods that are used to make the power-ups to work. One of the methods generates a random power-up type, the other creates the power-up based on the race and the last one checks if one of the cars picked it up. The method that creates a new power-up is optimized because we are getting the relevant checkpoints based on the race lvl so we are using an if statement. The method that checks if a car picked the power-up is optimized because we are getting the cardId as a parameter from the method definition and based on the car id we are checking if the car overlaps with the power-up and only in the case that it do we applying that power up to the relevant car.

For drawing the car health we are using a method based on the car position on the map the health bar and the healths points are shown on the racetrack. Another important method for the game is the one that shows the game screen. Here we are getting the user input (the arrow keys that the player press) and base on them we are interacting with the car. Inside this method we are also checking for the collision between the cars, we are making the cars to move and even draw them. Here is also happening the countdown that appears before the race starts and the race status like the place, time and lap, and even the mini-map.

In the code is a method that checks if a level is unlocked. This method is optimized because it returns true in the case that the level in the cause is lower than the total number of unlocked levels, otherwise it returns false. Another method is the one that draws the menu from the game.

This method is optimized because the buttons are loaded using a method from a different header file. The methods that are drawing the screen that contains the game instructions and the one where you can select your car are optimized because the assets are loaded using a method that loaded the assets only one time and recycles them after we don't use them anymore.

The methods that are used for the level selection screen are well optimized because based on the level number we are drawing using a method the buttons. At the game run, we are creating the sprites for the most used items in the game such as the racetrack and the cars.

9. Evaluation and future developments

Dangerous Racers is the second game that I finished and was written in C++ using the SFML library. At the beginning I was worried because I thought that I won't be able to create a game that contains cars that must go forward or backward, to turn around or even collide with other objects.

One of the biggest problems that I encountered while creating the game was how to make the cars to move automatically, but after a few tries I manage to do it faster than what I planned – all the car games have mechanics that are related to the natural science field, therefore it wasn't so hard to understand how the algorithm works.

Another problem was encountered when I wanted to create the minimap. Basically, the minimap represents the entire racing track but on a different scale. Based on this I manage to create the minimap successfully. A feature that was added in the last minute was the ability to pick power-ups. Even if there are only 4 power-ups in the entire game I think that they are very useful and enough for a 3 level game.

In the future versions, it would be important if the game will contain different type of racing: for instance one of the types will be to win the race and it will be the same as the game is now, another one is to escape from more police cars and not in the last a mode where you can race against your friends.

Also, I think that would be important if add some sounds into the game for instance when the countdown is, when a car accelerates or when they crash with each other when they explode or a sound that is always playing in the background.

10. References

[Fig. 1][Fig. 2][Fig. 3][Fig. 4]

Unlucky Studio. (2015). *Free Top Down Car Sprites*. Available: https://opengameart.org/content/free-top-down-car-sprites-by-unlucky-studio. Last accessed 24th Oct 2019.

[Fig. 5][Fig. 7][Fig. 9]

freepik. (2019). *Collection of f1 racing tracks.* Available: https://www.freepik.com/free-vector/collection-f1-racing-tracks_2775870.htm#page=1&query=race%20track&position=36. Last accessed 24th Oct 2019.