Documentation – Drag Racing

Introduction

The “Drag Racing” is a racing game that allows players to experience the thrill of drag racing. The game features one level and a variety of cars to choose from.

In the game, players must race their car down a straight track as fast as possible, competing against the other car and trying to beat it.

The Drag Racing game is simple which makes it easy for both beginners and experienced gamers to play. Drag Racing game provides entertainment for racing fans of all ages.

User Manual

Welcome to the Drag Racing! In this game, you will experience the thrill of drag racing, competing against other car on a straight track.

Getting Started:

To start the game, simply run the executable file or start the game from your Python development environment. Once the game has started, you will be presented with the car selection.

When you have selected your car, the game brings you to the track and the countdown for the race begins.

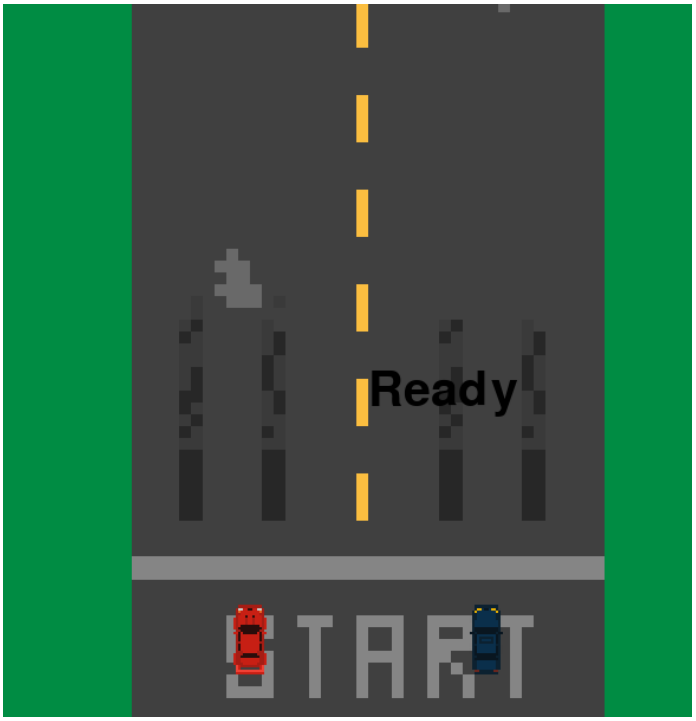
Car Selecting Menu

Controls:

Once you have started the game, you can control your car using the keys on your keyboard. The up arrow key accelerates your car, and space key uses nitrous

Conclusion:

That's it! You are now ready to play the Drag Racing game. Enjoy the game and good luck beating your opponent.



Code

Language

The game is built with Python programming language.

Classes

1. Camera

```
import pygame
import os


class Camera(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(os.path.join("images", "background.png"))
        self.image = pygame.transform.scale(self.image, (640,10000))
        self.rect = self.image.get_rect()
        self.rect.center = (320, -4300)
        self.camera_y = 0

    def control(self, speed):
            self.camera_y = speed

    def update(self):
        self.rect.y = self.rect.y + self.camera_y
```

2. Car

```
class Car:
    # initialize car class with model, image, speed,
    # make and nitrous_duration data attributes
    def __init__(self, ID,  make, model, speed, nitrous_duration, image):
        self.ID = ID
        self.make = make
        self.model = model
        self.speed = speed
        self.nitrous_duration = nitrous_duration
        self.image = image
```

3. Enemy

```
import os
import pygame
import cars_module

# create a enemy sprite class
class Enemy(pygame.sprite.Sprite):
    # enemy takes a position value and
    # chosen car as string
    # Initialize Player class with
    # data attributes chosen_car, image, rect and speed
    def __init__(self, position, chosen_car):
        #player inherits sprite
        pygame.sprite.Sprite.__init__(self)

        #set attribute chosen_car's value to chosen car from cars_module
        self.chosen_car = cars_module.choose_car(chosen_car)

        #use os to join folder name to chosen car objects image value
        self.image = pygame.image.load(os.path.join("images",\
            getattr(self.chosen_car, "image")))

        self.image = pygame.transform.scale(self.image, (28,59))
        self.rect = self.image.get_rect()
        self.rect.center = position
        #set speed to chosen car objects image value
        self.speed = getattr(self.chosen_car, "speed")
        self.nitrous_duration = getattr(self.chosen_car, "nitrous_duration")


    def update(self):
        self.rect.y = self.rect.y - self.speed

    def get_speed(self):
        return self.speed

    def get_y(self):
        return self.rect.y
```

4. Goal

```
import pygame
import os


class Goal(pygame.sprite.Sprite):
    def __init__(self):
        pygame.sprite.Sprite.__init__(self)
        self.image = pygame.image.load(os.path.join("images", "goal.png"))
        self.rect = self.image.get_rect()
        self.rect.center = (320, -4300)
        self.goal_y = 0

    def control(self, speed):
        self.goal_y = speed * 18
        print(self.rect)

    def update(self):
        self.rect.y = self.rect.y + self.goal_y
```

5. Player

```
import os
import pygame
import cars_module

# create a player sprite class
class Player(pygame.sprite.Sprite):
    # Player takes a position value and
    # chosen car as string
    # Initialize Player class with
    # data attributes chosen_car, image, rect and speed
    def __init__(self, position, chosen_car):
        #player inherits sprite
        pygame.sprite.Sprite.__init__(self)

        #set attribute chosen_car's value to chosen car from cars_module
        self.chosen_car = cars_module.choose_car(chosen_car)

        #use os to join folder name to chosen car objects image value
        self.image = pygame.image.load(os.path.join("images",\
            getattr(self.chosen_car, "image")))

        self.image = pygame.transform.scale(self.image, (28,59))
        self.rect = self.image.get_rect()
        self.rect.center = position
        #set speed to chosen car objects image value
        self.speed = getattr(self.chosen_car, "speed")
        self.movey = 0
        self.nitrous_duration = getattr(self.chosen_car, "nitrous_duration")
        self.nitrous_on = False
        self.make = getattr(self.chosen_car, "make")
        self.model = getattr(self.chosen_car, "model")

    # set players movey value to speed of the instance of car class
    def control(self, speed):
        self.movey = - speed

    def set_nitrous_on(self, set_nitrous_on):
            self.nitrous_on = set_nitrous_on


    #update player position. if nitrous is one
    def update(self):
        self.rect.y = self.rect.y + self.movey
        if self.nitrous_on and (self.nitrous_duration > 1):

            self.rect.y = self.rect.y + self.movey-3
            print(self.nitrous_duration)
            self.nitrous_duration -= 1
```

```
    #return the speed value of the instance of car class
    def get_speed(self):
        return self.speed

    def get_y(self):
        return self.rect.y

    #return car make + model for menu
    def get_car_name(self):
        return self.make + " " + self.model
```

Test report: Drag Racing

Tested by: Veeti Virtanen

Date 19.4.2023

Computer Used for Testing:

Model: Legion T5 26AMR5, AMD Ryzen 5 5600G CPU, NVIDIA GeForce GTX 3060 16GB RAM GPU

Screen: Zowie by Benq XL2540K, 240Hz refresh rate

Controller: Glorious GMMK PRO Keyboard

Testing Process:

The testing process includes the following steps:

1. **Installation**: The game was installed on Windows 11 PC and launched to make sure it runs correctly.

2. **Unit Testing**: Each module of the game was tested to make sure everything works correctly and produces the expected output.

3. **User Testing**: The game was tested by users to identify any bugs or issues that need to be fixed, such as crashes or graphical glitches.

Testing Results

1. **Installation**: The game installed correctly and ran without any issues on every PC

2. **Unit Testing:** All tested modules functioned correctly and produced the expected output.

3. **User testing:** The game worked like it is supposed to work and did not have any bugs or issues.

System requirements

Operating system: Windows 7 (64bit) or later

Processor: Intel Core 2 Duo E5200

Memory: 4 GB RAM

Storage: 5 GB