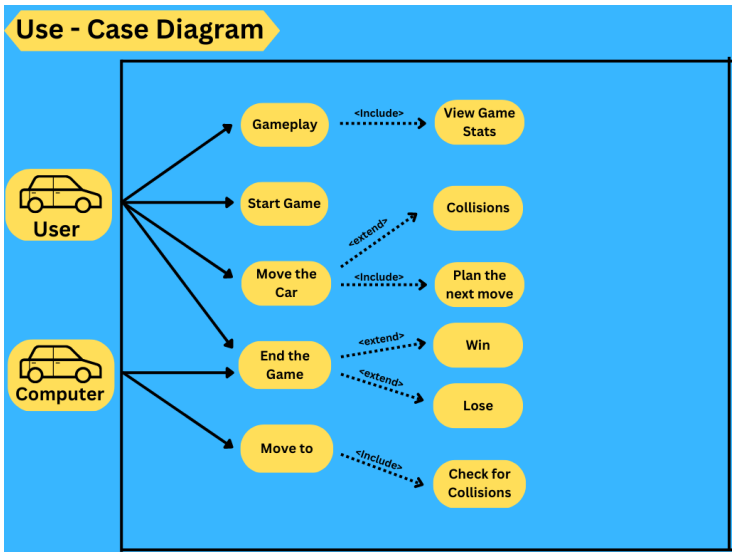
Racing Game

A. Project documentation :

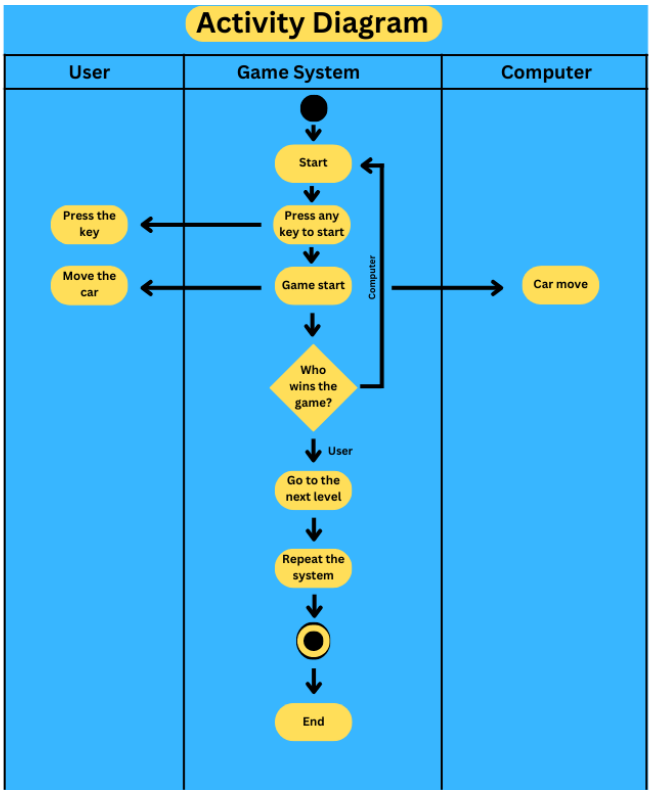
Brief description :

This racing game is inspired by a game called super pixel racers which was originally made for nintendo. However , I made some changes for this game so the player only has to race the computer's car. Which makes it easy for all of us. In addition, I also add levels to the game to make the game more challenging and also captivating. The track in this game is pretty straightforward and there is only 1 track available. The source code is from a youtuber named "Tech with Tim".

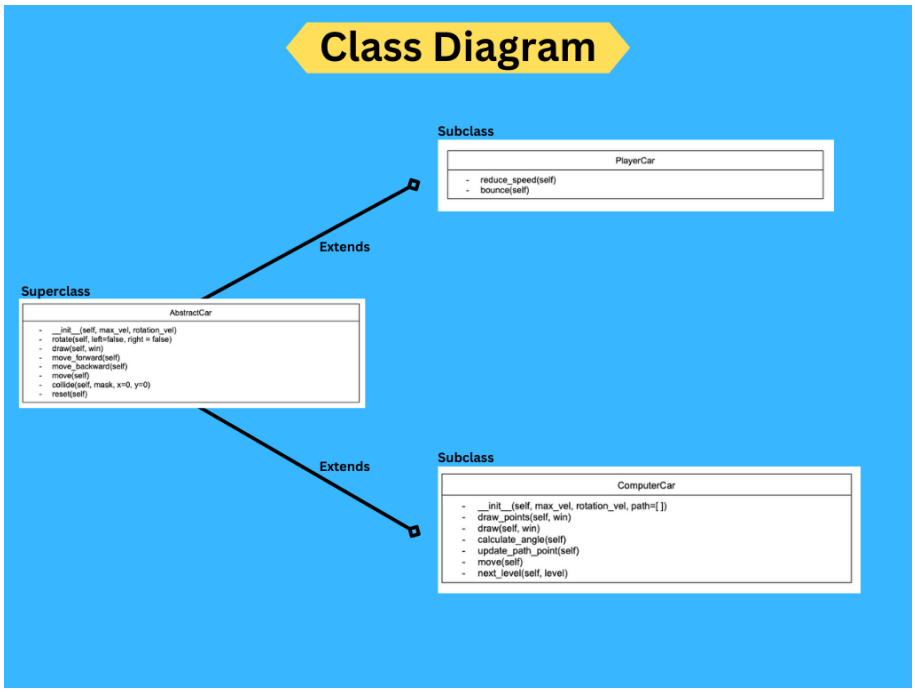
Use - Case Diagram :



Activity Diagram :



Class Diagram :



Modules that we used :

Pygame.display.update - to update the display window and the screen of the game
Pygame.display.set_mode - initialize a window or a screen for the display of the game
Pygame.display.set_caption - set the title of the window (in top of the window)
Pygame.key - to assigned the key to the game
Pygame.key.get_pressed - returns a list with the state of each key
Pygame.time.wait - pause the program for certain time
Pygame.time.clock - create a clock object to track an amount of time
Pygame.event.get - to get the events from the queue
Pygame.quit - to exit the pygame window/the game
Pygame.keydown - indicating if the key is pressed or not
Pygame.font - for load and rendering the font that we applied
Pygame.font.Sysfont - create a font object from the system fonts
Pygame.image.load - load a new image from a file
Pygame.mask.from_surface - create a mask from a surface that we choose
Pygame.draw.circle - to draw a circle with the size that we want
Pygame.Rect - to store and manipulate rectangular areas
Pygame.K_a - to know that the key A is pressed
Pygame.K_d - to know that the key D is pressed
Pygame.K_s - to know that the key S is pressed
Pygame.K_w - to know that the key W is pressed

Essential Algorithm :

```
def calculate_angle(self):                                  # calcula
    target_x, target_y = self.path[self.current_point]      # define
    x_diff = target_x - self.x                              # calcult
    y_diff = target_y - self.y                              # calcula

    if y_diff == 0:                                         # the dis
        desired_radian_angle = math.pi / 2                  # the ang
    else:
        desired_radian_angle = math.atan(x_diff / y_diff)   # the ang

    if target_y > self.y:
        desired_radian_angle += math.pi                     # to make

    difference_in_angle = self.angle - math.degrees(desired_radian_angle)   #
    if difference_in_angle >= 180:                                          #
        difference_in_angle -= 360                                          #

    if difference_in_angle > 0:
        self.angle -= min(self.rotation_vel, abs(difference_in_angle))      #
    else:
        self.angle += min(self.rotation_vel, abs(difference_in_angle))      #

def update_path_point(self):                                                #
    target = self.path[self.current_point]                                  #
    rect = pygame.Rect(                                                     #
        self.x, self.y, self.img.get_width(), self.img.get_height())       #
    if rect.collidepoint(*target):                                          #
        self.current_point += 1                                             #

def move(self):                                                             #
    if self.current_point >= len(self.path):                                #
        return

    self.calculate_angle()                                  # calcula
    self.update_path_point()                                # update
    super().move()                                          # call th

def next_level(self, level):                                # reset t
    self.reset()                                            # reser t
    self.vel = self.max_vel + (level - 1) * 0.2             # define
    self.current_point = 0
```

```
def move_player(player_car):
    keys = pygame.key.get_pressed()
    moved = False

    if keys[pygame.K_a]:
        player_car.rotate(left=True)
    if keys[pygame.K_d]:
        player_car.rotate(right=True)
    if keys[pygame.K_w]:
        moved = True
        player_car.move_forward()
    if keys[pygame.K_s]:
        moved = True
        player_car.move_backward()

    if not moved:
        player_car.reduce_speed()
```

```python
class AbstractCar:                                          # class to mov
    def __init__(self, max_vel, rotation_vel):              # define the m
        self.img = self.IMG
        self.max_vel = max_vel
        self.vel = 0                                        # initial spee
        self.rotation_vel = rotation_vel
        self.angle = 0                                      # the car is s
        self.x, self.y = self.START_POS
        self.acceleration = 0.1                             # everytime th

    def rotate(self, left=False, right=False):              # initialize t
        if left:
            self.angle += self.rotation_vel                 # if the car m
        elif right:
            self.angle -= self.rotation_vel                 # if the car r

    def draw(self, win):
        blit_rotate_center(win, self.img, (self.x, self.y), self.angle)

    def move_forward(self):
        self.vel = min(self.vel + self.acceleration, self.max_vel)
        self.move()

    def move_backward(self):
        self.vel = max(self.vel - self.acceleration, -self.max_vel/2)
        self.move()

    def move(self):
        radians = math.radians(self.angle)
        vertical = math.cos(radians) * self.vel
        horizontal = math.sin(radians) * self.vel

        self.y -= vertical
        self.x -= horizontal

    def collide(self, mask, x=0, y=0):                      # function
        car_mask = pygame.mask.from_surface(self.img)
        offset = (int(self.x - x), int(self.y - y))         # substrac
        poi = mask.overlap(car_mask, offset)                # return t
        return poi

    def reset(self):                                        # reset th
        self.x, self.y = self.START_POS                     # initiali
        self.angle = 0                                      # set the
        self.vel = self.max_vel                             # the car
        self.current_point = 0
```

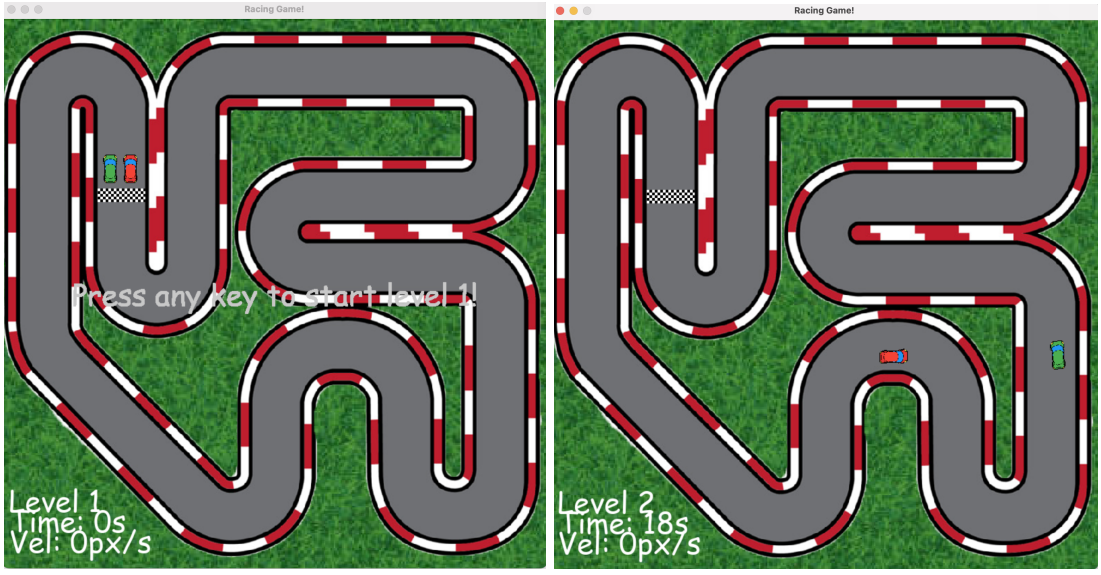
```python
def scale_image(img, factor):
    size = round(img.get_width() * factor), round(img.get_height() * factor)
    return pygame.transform.scale(img, size)


def blit_rotate_center(win, image, top_left, angle):
    rotated_image = pygame.transform.rotate(image, angle)
    new_rect = rotated_image.get_rect(
        center=image.get_rect(topleft=top_left).center)
    win.blit(rotated_image, new_rect.topleft)


def blit_text_center(win, font, text):
    render = font.render(text, 1, (200, 200, 200))
    win.blit(render, (win.get_width()/2 - render.get_width() /
                      2, win.get_height()/2 - render.get_height()/2))
```

```python
class GameInfo:
    LEVELS = 10

    def __init__(self, level=1):
        self.level = level
        self.started = False
        self.level_start_time = 0

    def next_level(self):
        self.level += 1
        self.started = False

    def reset(self):
        self.level = 1
        self.started = False
        self.level_start_time = 0

    def game_finished(self):
        return self.level > self.LEVELS

    def start_level(self):
        self.started = True
        self.level_start_time = time.time()

    def get_level_time(self):
        if not self.started:
            return 0
        return round(time.time() - self.level_start_time)
```

Lessons learned :

As a student, I learned a lot to make this game, especially to understand each module of pygame that is available. Hundreds of trial and error, but somehow I successfully made it. Building this game might not be difficult for an expert, but as a beginner, I am really proud of myself since this is my first game and I can't wait to present it to my classmates. This might be a simple racing game but my effort to build this game is a lot. And from that, I learned that time management in making the project is really hard. We also have to know the risk of making this project and what we gain from it.

Screenshots of our applications :



Reference:

Tech With Tim. (2021, October 8). *Pygame Car Racing Tutorial #1 - Moving The Car* [Video]. YouTube. https://www.youtube.com/watch?v=L3ktUWfAMPg

Tech With Tim. (2021b, October 12). *Pygame Car Racing Tutorial #2 - Pixel Perfect Collision* [Video]. YouTube. https://www.youtube.com/watch?v=WfqXcyF0_b0

Tech With Tim. (2021c, October 14). *Pygame Car Racing Tutorial #3 - Computer Car And Path Following* [Video]. YouTube. https://www.youtube.com/watch?v=V_B5ZCli-rA

Tech With Tim. (2021d, October 16). *Pygame Car Racing Tutorial #4 - Levels, Main Menu and Finishing Touches* [Video]. YouTube. https://www.youtube.com/watch?v=agnVylvZ038

Python Module Index — pygame v2.1.4 documentation. (n.d.). https://www.pygame.org/docs/py-modindex.html