Code

```
import math


def dist(point1, point2):
  return ((point1[0] - point2[0]) ** 2 + (point1[1] - point2[1]) ** 2) ** 0.5


def rect(r, theta):
  x = r * math.cos(math.radians(theta))
  y = r * math.sin(math.radians(theta))
  return x, y

def polar(x, y):

  r = (x ** 2 + y ** 2) ** .5
  theta = math.degrees(math.atan2(y,x))
  return r, theta

def angle_mod_360(angle):
  n = math.floor(angle/360.0)
  angle_between_0_and_360 = angle - n*360.0

  if angle_between_0_and_360 <= 180.0:
    return angle_between_0_and_360
  else:
    return angle_between_0_and_360 - 360

def get_waypoints_ordered_in_driving_direction(params):
  if params['is_reversed']:
    return list(reversed(params['waypoints']))
  else:
    return params['waypoints']


def up_sample(waypoints, factor):
  p = waypoints
  n = len(p)

  return [[i / factor * p[(j+1) % n][0] + (1 - i / factor) * p[j][0],
```

```
        i / factor * p[(j+1) % n][1] + (1 - i / factor) * p[j][1]] for j in range(n) for i
in range(factor)]

def get_target_point(params):
  waypoints = up_sample(get_waypoints_ordered_in_driving_direction(params),
20)

  car = [params['x'], params['y']]

  distances = [dist(p, car) for p in waypoints]
  min_dist = min(distances)
  i_closest = distances.index(min_dist)

  n = len(waypoints)

  waypoints_starting_with_closest = [waypoints[(i+i_closest) % n] for i in
range(n)]

  r = params['track_width'] * 0.9

  is_inside = [dist(p, car) < r for p in waypoints_starting_with_closest]
  i_first_outside = is_inside.index(False)

  if i_first_outside < 0:
    return waypoints[i_closest]

  return waypoints_starting_with_closest[i_first_outside]

def get_target_steering_degree(params):
  tx, ty = get_target_point(params)
  car_x = params['x']
  car_y = params['y']
  dx = tx-car_x
  dy = ty-car_y
  heading = params['heading']

  _, target_angle = polar(dx, dy)

  steering_angle = target_angle - heading

  return angle_mod_360(steering_angle)
```

```
def score_steer_to_point_ahead(params):
  best_stearing_angle = get_target_steering_degree(params)
  steering_angle = params['steering_angle']
  error = (steering_angle - best_stearing_angle) / 60.0

  score = 1.0 - abs(error)

  return max(score, 0.01)


def reward_function(params):
  return float(score_steer_to_point_ahead(params))
```