CMPM 146

Assignment 1: Steering Seek

Calvin Li & Luis Rocha

April 17, 2025

Assignment Questions:

1. How do you determine which direction to turn?

In order to determine which direction to turn, inside of the SeekTarget() method, we use the Vector3.SignedAngle() Function:

```
float angle = Vector3.SignedAngle(forward, dirToTarget, Vector3.up);
```

This function calculates the signed angle (positive or negative) between the cars forward direction (transform.forward) and the direction towards the target (dirToTarget). In our case the Y-axis is the rotation axis.

If the angle is positive then the target is to the right of the car's current direction, so the car turns right. Vice versa, if the angle is negative the target is to the left, so the car turns left. If the angle is close to zero then there is no need to turn since it's already facing the target.

In order to know how much and how quickly to rotate, we use:

```
float rotationSpeed = angleAbs < 2f ? 0f :
Mathf.Clamp(angle * rotationT * 2f, -kinematic.GetMaxRotationalVelocity(),
kinematic.GetMaxRotationalVelocity());
```

This code allows the car to turn proportionally based on both the angle to the target and the distance (via rotationT).

2. Did you change any of the car’s parameters? (max speed, acceleration, etc?)

No, we did not change any of the car's parameters, we found the presets were adequate for fluid movement.

3. How do you slow down as you approach the target?

To slow down as we approach the target, we used the distance to the target in order to scale the speed down. This was done calculating the normalized value “SpeedT” and a new variable “SlowingDistance”

```
float speedT = Mathf.Clamp01(distance / slowingDistance);

float speedFactor = Mathf.SmoothStep(0f, 1f, speedT);
```

Since the car is non-mono, it cannot rotate on the spot, so we combined this slowingDistance value with angleFactor to slow down more if the car needs to change direction.

```
float angleFactor = Mathf.Clamp01((180f - angleAbs) / 180f);

float desiredSpeed = kinematic.GetMaxSpeed() * speedFactor * angleFactor;
```

This desiredSpeed is input to Kinematric.SetDesiredSpeed to set the cars speed at any given time.

This desired speed stops the car from overshooting by having too much velocity. We also used arrivalRadius in order to allow the car to move almost to a stop when it gets close enough.

4. When following a path, how do you make sure to turn smoothly around corners?

So to turn smoothly, we had a variable called “A.”, which is like a lookahead for the next waypoint to see the distance between the car and the next waypoint, and through that we were able to detect when the car should turn. Later in the code, we have an if “B” to turn sharper corners. Since this was just to see the waypoint and not how we actually did a turn, we instead had to add angles, and depending on the angle, we decided if the car should turn earlier or not. This was later down in the code, where the angle degree was important. Even with the angles, the car would still overshoot sometimes and make a small arc to get back into the path.

A.float advanceRadius = 20f;

B. if (angle between < 70f) advanceRadius = 22f;

5. Did you encounter any particular challenges during the assignment?

Some challenges that we had early on during the seek target were adjusting the number correctly for the car to stop at or close to the point, but we eventually figured it out. For part 2, we had the same challenge but with angels instead. After figuring out how to turn smoothly, we also had to figure out if a angle was negative(do a reverse backwards), a sharp turn(slow down before the point and turn slowly), and we couldn't really figure out how to make it so it doesn't overshoot sometimes and make a big arc/180 turn to get to the next point. Sometimes the car would overshoot at the end, but I just figured to put a little reverse at the end so it readjusts itself at the actual point.

Individual Retrospective:
What did you contribute to the assignment, and what did you learn?

Both partners contributed to this document evenly.

Calvin Li:
I began by expanding on the part 1 code by working on fluid arrival at the target. At first, we had an issue where the car would go in slow circles around the target and would not stop properly. I implemented an arrival radius that slowed the car down and arrived on or near the target. I also fine-tuned the distance and angle-based speed scaling to prevent sudden stops or oversteering. I expanded on the part 1 code for the arrival radius so that the car would slow down as the car neared each point without having to directly touch each one while going to the next. I also added dynamic speed control that would depend on the sharpness of the next turn so that corners would transition smoothly. This prevents the car from stopping hard at each point in its path. Through part 1, I learned how to combine Unity Vector math with the non-holonomic movement to produce smooth and more realistic driving physics. Moreover, in part 2, I learned how to make paths look smooth so the car did not stop and start at each point.

Luis Rocha:
In this assignment, I began by working on Part 1, My main focus was to figure out the correct direction to turn the car based on where the target was using Vector3.SignedAngle. I fine-tuned the handling of rotational Velocity and angle-based rotation so that the car turns correctly and does not have jittery movements. For part 2, I handled the path following logic and how the car would go from one waypoint to the next. Worked with Calvin to work on predictive turning so the car rotates before it reaches the next waypoint, but while at least close to the target. Finally, I made sure that the car would do the complete path in the correct order it was created. From part 1, I learned how to use the positive/negative angles from Vector3.SignedAngle, so the car knew what direction to turn in for the waypoint. Through part 2, I learned how to connect waypoints

and create steering behaviour that does not cause the car to directly touch each waypoint when moving to the next.

AI acknowledgement:
We did not use any AI in the creation of this Assignment code or Report