RCAP CoSpace Autonomous Driving
(Useful Functions 2)

A POSTERIORI
Play · Experience · Learn

Competition Timeline

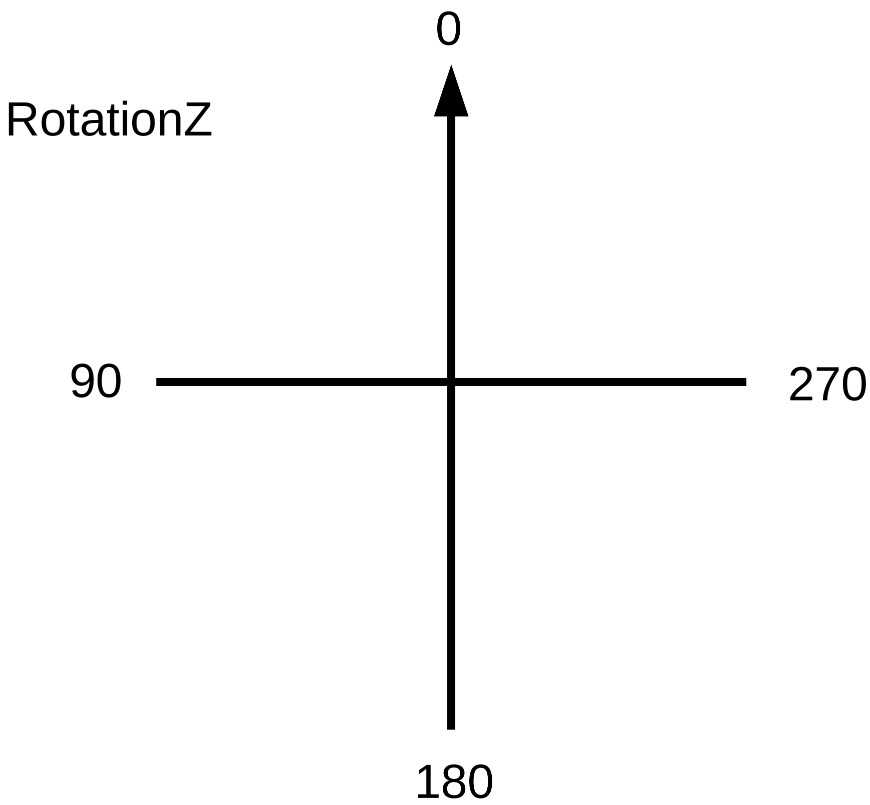
- 22 May (Team Description and Video)
  - Submit Team Description Paper & Video
  - Template will be provided by email
- 23 to 26 May (Warm-up)
  - Warm up exercises (...not graded)
  - Helps you familiarize yourself with competition procedure
- 29 May (Preliminary games) (Saturday)
  - Given a fixed time to solve challenge map
  - Do from home
  - Details to be sent via email

Competition Timeline

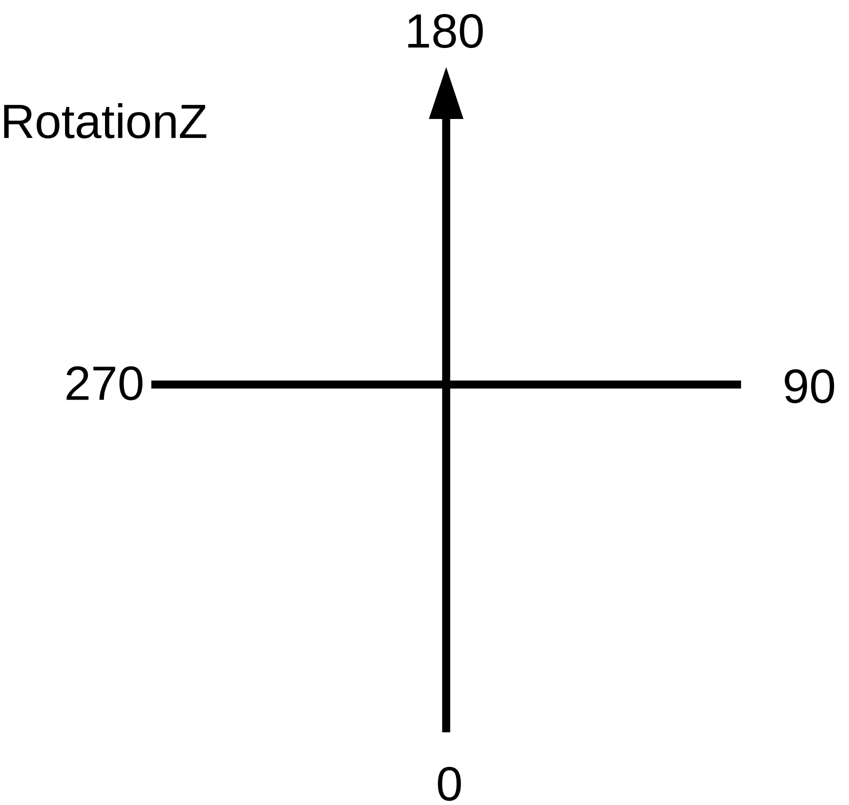
- 31 May (Announcement of Finalist)
  - Notified via email
- Finalists: 3 Jun (Video submission)
  - Another video. This time describing the game strategy
- Finalists: 5 - 9 Jun (Interview)
  - Interview via Zoom
- Finalists: 10 Jun (Announcement of selected students)
- Finalist: 12 Jun (Grand Finals)

Before We Start

The angles in Cospace are <u>worse</u> than I thought...



<u>Task 1 to 3 Maps</u>



<u>Future City Map</u>

Angles in Cospace

- Different across maps
  - Some “North = 0 degrees”
  - Others “North = 180 degrees”
  - Doesn’t depend on the direction the robot face at the start
  - Check at the start and modify your program accordingly
- So far, all the angles increases when rotating counter-clockwise
  - If this is not true, you’ll need to change the direction that your robot turns in Gyro follower

What Else?

- Actions
  - move_steering(steering, speed)
    - Separates “steering” (curvature of turn) and “speed”
    - Allows you to change speed without changing how much the robot turns
    - You may already have the algorithm (...it’s in the proportional line follower)
  - gyro_follow(angle, speed)
    - Used when not following line
    - Situational; May be useful for shortcuts
  - turn_to_angle(angle)
    - Turns fast at start, then slow down when close to angle

Why?

- Turn fast at the start
  - Save time
- Slows down near target angle
  - Accurate turns

Turn to Angle

- Very similar to line and Gyro follower

Line / Gyro Follower

- Look at line position or gyro angle
- Decide whether to...
  - Turn left
  - Turn right
  - Go straight

Turn to Angle

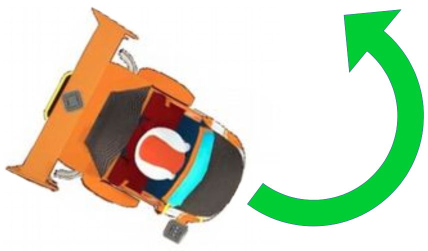
- Look at gyro angle
- Decide...
  - How fast to turn

- Main difference; Instead of controlling direction to move (**steering**), we control the **speed**

Turn to Angle

- There is already one provided!
  - It's called “TurnTo”, and it's in the default C source file
  - It uses a 5-states algorithm
  - Only does an on-the-spot turn (ie. no curve turn)

on-the-spot turn



curve turn

Proportional Control

1) Calculate the error
(error = whatYouHave – whatYouWant)

2) Calculate the correction
(correction = error * gain)

3) Apply the correction

Proportional Turn to Angle

1) Modify RotationZ so that angles on the right are always smaller and left always are larger (...this was covered last lesson)

2) Calculate the <u>error</u>
(error = what_you_have – what_you_want)

3) Calculate the <u>correction</u> (speed)
(speed = error * gain)

4) Use move steering to apply the speed

Proportional Turn to Angle

```
void turnToAngle(int angle, int steering)
{
  if (angle > 180) {
    if (RotationZ < (angle - 180)) {
      RotationZ += 360;
    }
  } else {
    if (RotationZ > (angle + 180)) {
      RotationZ -= 360;
    }
  }

  float err = RotationZ - angle;
  float speed = err * 0.3;
  moveSteering(speed, steering);
}
```

Modify RotationZ if necessary (...same as last week)

Calculate error

Calculate speed (gain = 0.3)

Apply speed to move steering

This function lets us set steering, so that we can use it for an on-the-spot turn, or for a curve turn.

Tuning Gain

- High Gain
  - Turns fast, but may overshoot and turn back

Overshoot and turn back

- Low Gain
  - Less overshoot, but turns slow
  - WARNING! If gain is too low, it may never reach the target angle

- We want high gain (...to be fast), but avoid the overshoot. How?

Derivative Control

- Proportional control looks at...
  - Position (eg. line position)
    ...or...
  - Angle (degrees)

- Derivative control looks at the rate of change...
  - Rate of change of position
  - Rate of change of angle (degrees per second)

Proportional and Derivative

Example: turnToAngle(90)

```
float err = RotationZ - angle;
float speed = err * 0.3;
```

- Proportional control:
  - We want angle to be 90 degrees.
  - If it is not, apply a correction

- Derivative control:
  - We want rate of turn to be 0 degrees per second
  - If it is not, apply a correction

Getting rate of change

- Angle is available from RotationZ
- What about rate of change of angle?

- Apply your math:
  - If the angle is 10 degrees at t=0, and 25 degrees at t=2, what is the rate of change of angle?
  - rate of change = (end – start) / time
    = (25 – 10) / 2
    = 15 / 2
    = 7.5 degrees per second

Getting rate of change

rate of change = (end – start) / time

static means that the variable will only be set the first time the function is executed

We initialize it to an impossible angle at the start

If the angle is 1000, this means we don’t have an actual previous angle yet, so we can’t calculate the rate.

```
static int prevAngle = 1000;
float rateOfAngle = 0;

if (prevAngle != 1000) {
  rateOfAngle = (RotationZ - prevAngle) / 0.025;
}
prevAngle = RotationZ;
```

Save the current rotation.

Calculate rate using the formula

Derivative Control

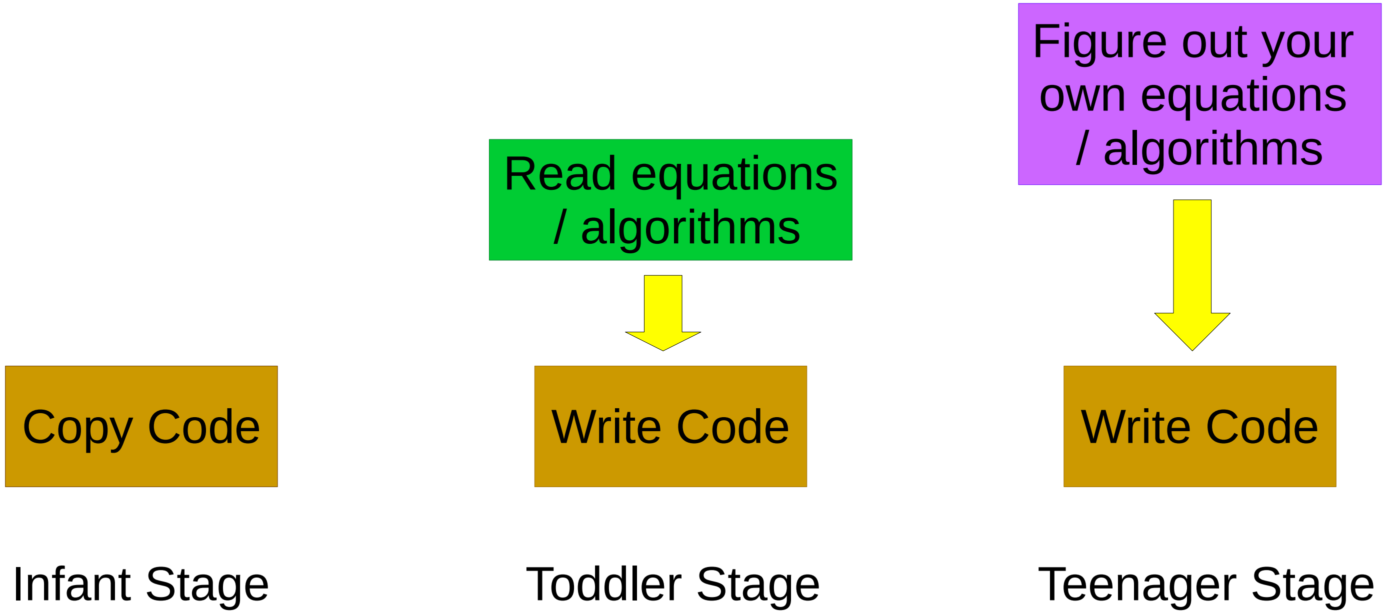
1) Calculate the error
   (d_error = whatYouHave – whatYouWant)
   (d_error = rateOfAngle – 0) // We want rate to be 0
   (d_error = rateOfAngle)

2) Calculate the correction
   (d_correction = d_error * d_gain)
   (d_speed = rateOfAngle * -1)
   Notice that d_gain is negative because we want to slow down

3) Combine with Proportional control and apply the correction
   move_steering(p_speed + d_speed, steering)

Code?

- Nope. That’s for you to figure out.
- I’ve already covered all the tricky bits.
- You won’t learn if you’re just copying code.



Copyright

A POSTERIORI

Play · Experience · Learn