

Simulation

Mobile Robotics

Anthony J. Clark

Today

- Simulation
- Second set of slides for motion planning

Simulation

- Find the right level of abstraction
- Answer the question: what do you care about?

Simulation

- Find the right level of abstraction
- Answer the question: what do you care about?
 - Do you care about the electromechanical properties?
 - Do you care about inertia?
 - Do you care about battery levels or capacity?
 - Do you care about noise? Or wheel slippage?

Analytical Model Simulation

Example: A Two-Dimensional WMR

$$x_t = x_0 + \omega r t$$

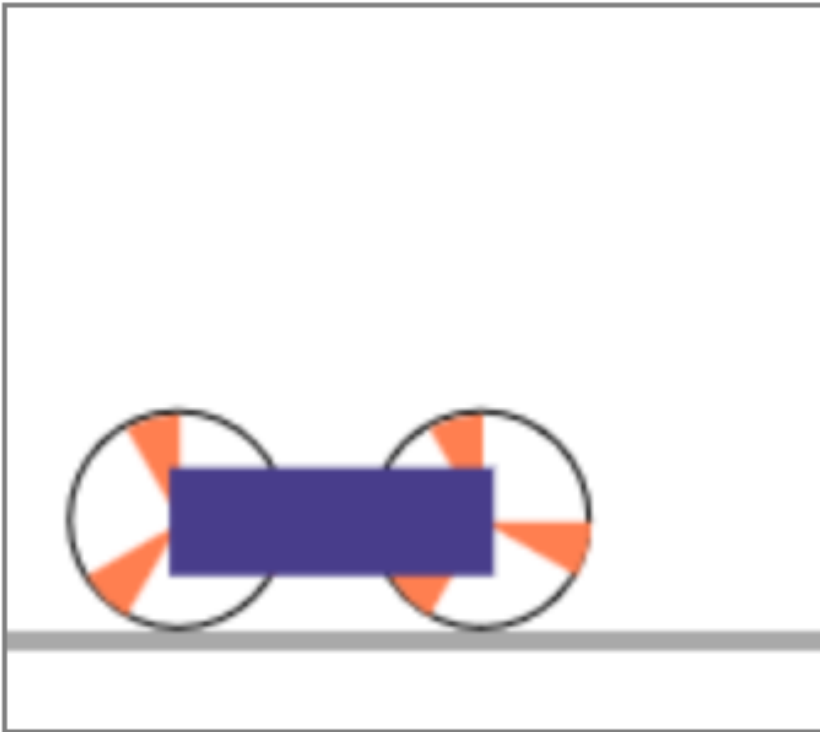


[Analytical Model Simulation](#)

Analytical Model Simulation

Example: A Two-Dimensional WMR

$$x_t = x_0 + \omega r t$$



A diagram of a two-dimensional wheeled mobile robot (WMR) on a horizontal surface. The robot consists of a central purple rectangular chassis and two circular wheels on either side. Each wheel has a white outer ring and four orange triangular segments. The wheels are in contact with a gray horizontal line representing the ground.

```
let initialPosition = 0;
let chassisPosition;

let angularVelocity = 1;
let wheelRadius = 1;

function simulate( time ) {

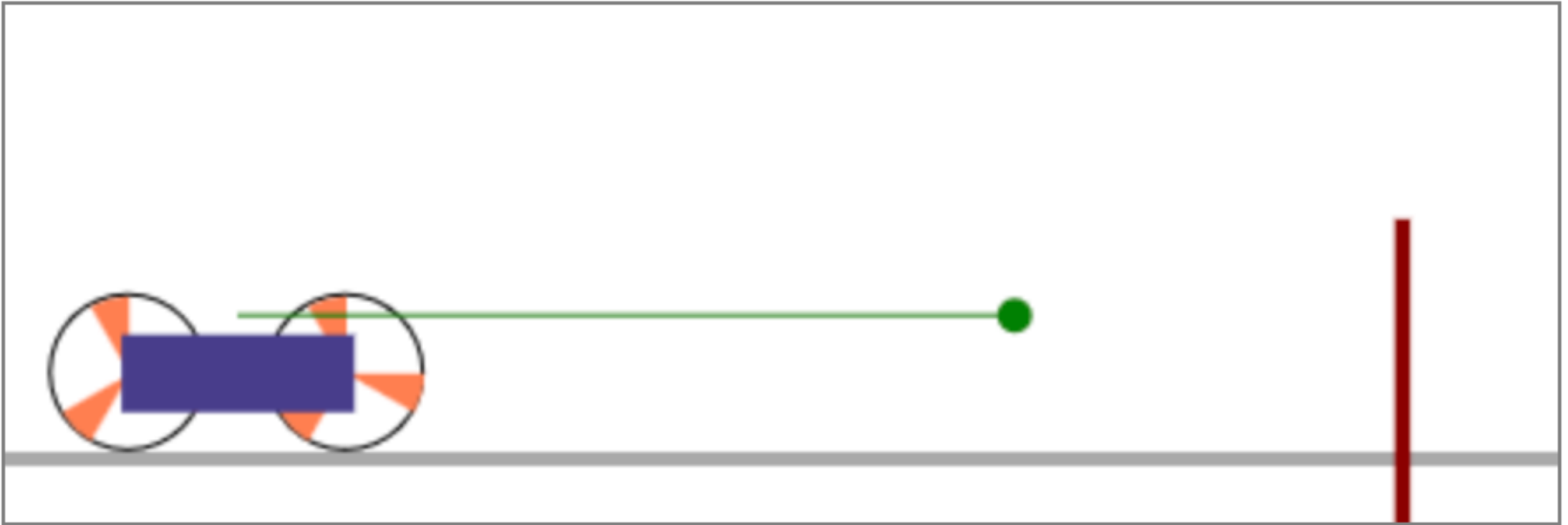
    // x = x0 + ω r t
    v = angularVelocity * wheelRadius
    chassisPosition = initialPosition + v * time;

}
```

[Analytical Model Simulation](#)

What if We Add a Wall?

$$x_t = x_0 + \omega r t$$



[Analytical Model Simulation](#)

What if We Add a Wall?

$$x_t = x_0 + \omega r t$$

```
if ( distanceToWall > 0 ) {  
    chassisPosition = initialPosition + v * time;  
}
```



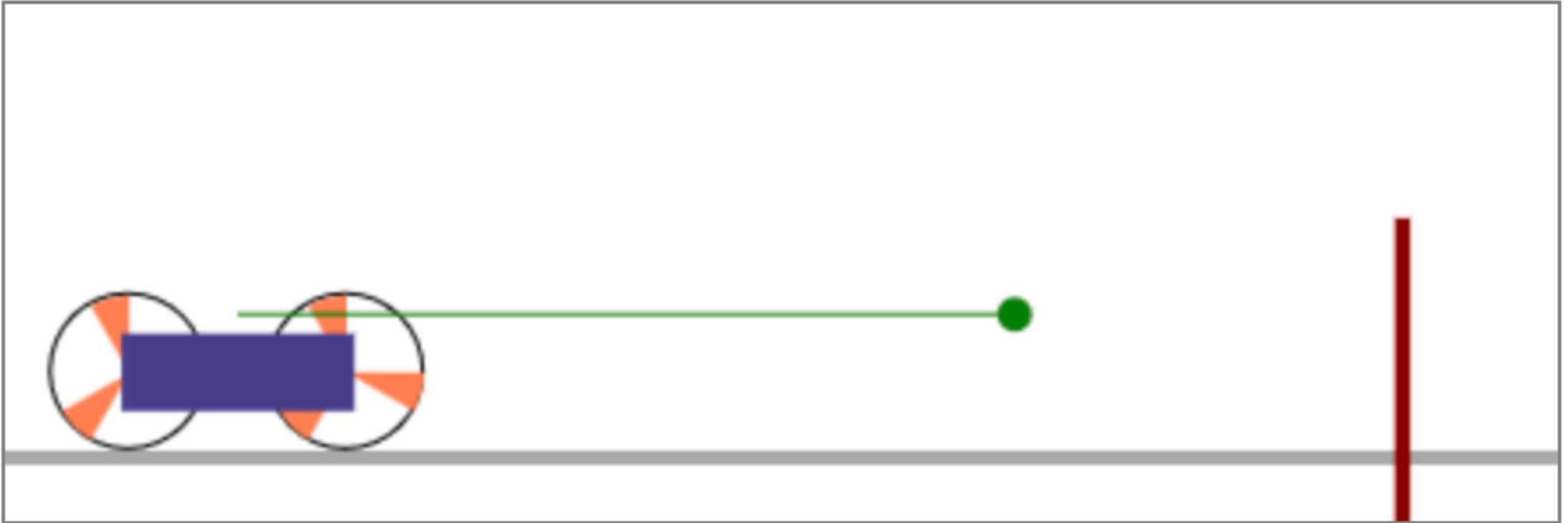
[Analytical Model Simulation](#)

What if We Want Dynamic Control?

$$x_t = x_0 + \omega r t$$

Where,

$$\omega = \text{constrain}(Kd + b, -\Omega, \Omega)$$

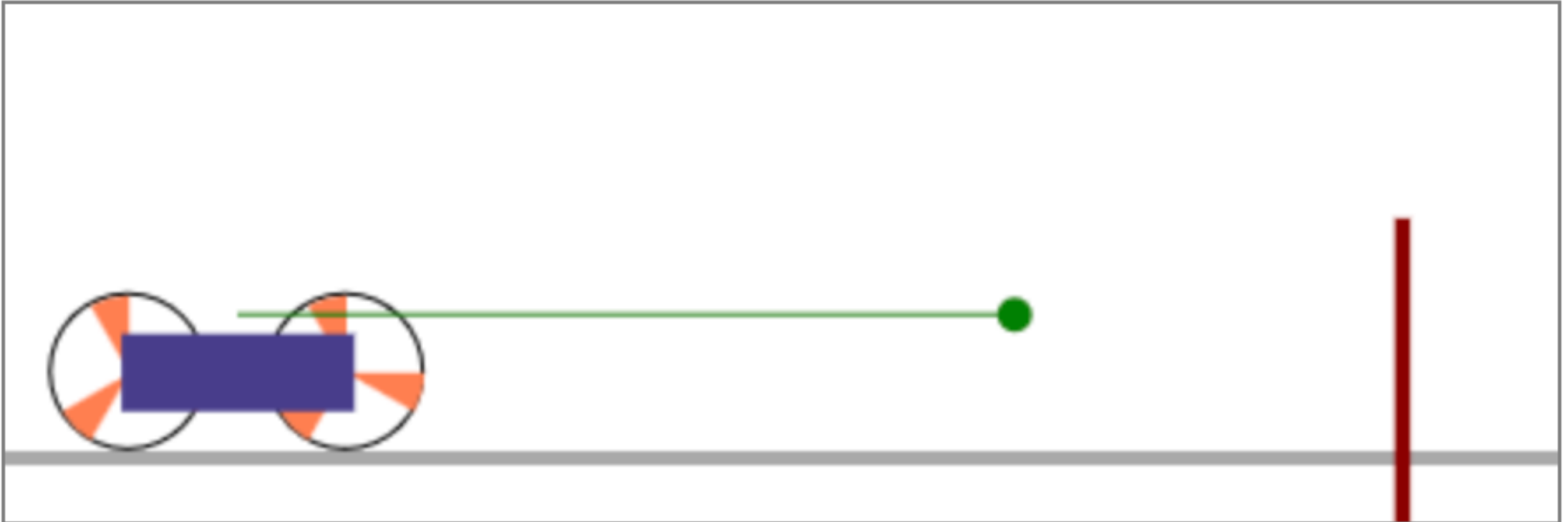


Analytical Model Simulation

Numerical Simulation

$$x_t = x_0 + \omega r t \quad (1) \text{ Analytical}$$

$$x_t = x_{t-1} + \omega r \Delta t \quad (2) \text{ Numerical}$$



Dynamic Control

Numerical Simulation Code

$$x_t = x_{t-1} + \omega r \Delta t$$

```
let initialPosition = 0;  
let chassisPosition = initialPosition;  
  
let angularVelocity = 1;  
let wheelRadius = 1;
```

```
function simulate( duration ) {  
  
    let time = 0;  
    const timeStep = 0.01;  
    let v = angularVelocity * wheelRadius  
  
    while ( time < duration ) {  
  
        chassisPosition += v * timeStep;  
        time += ;  
  
    }  
  
}
```

Demo