ROBOT BEHAVIOR

Maslab ‘09

Overview

- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

Overview

- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

The Big Picture… Literally

Overview

- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

Your Goal

- Find balls
- Pick up balls
- Dump balls in goals

Your Goal

- Find balls
- Pick up balls
- Dump balls in goals
- Navigate the robot out if its corner, through the small gap

Your Goal

- Find balls
- Pick up balls
- Dump balls in goals
- Navigate the robot out if its corner, through the small gap
- Don’t overload robot

Your Goal

- Find balls
- Pick up balls
- Dump balls in goals
- Navigate the robot out if its corner, through the small gap
- Don’t overload robot
- Avoid walls

The Game

- More complicated than just picking up balls
- Find all the caveats at:
  - http://maslab.mit.edu/2009/wiki/Playing_field
  - http://maslab.mit.edu/2009/wiki/Rules

Overview

- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

The Wrong Way – An Example

```
void moveForward( int time ) {

  while ( t < time ) {

    // Drive forward a bit
    ---------------------------------------
    ---------------------------------------

  }
}
```

The Wrong Way – An Example

```
void moveForward( int time ) {

  while ( t < time ) {

    // Drive forward a bit
    ---------------------------------------
    ---------------------------------------

    // Check ir sensor and stop if necessary
    ---------------------------------------
    ---------------------------------------

  }
}
```

The Wrong Way – An Example

```
void moveForward( int time ) {

  while ( t < time ) {

    // Drive forward a bit
    ---------------------------------------
    ---------------------------------------

    // Check ir sensor and stop if necessary
    ---------------------------------------
    ---------------------------------------

    // Rotate if there is an obstacle
    ---------------------------------------
    ---------------------------------------

  }
}
```

The Wrong Way – An Example

```
void moveForward( int time ) {

  while ( t < time ) {

    // Drive forward a bit
    ---------------------------------------
    ---------------------------------------

    // Check ir sensor and stop if necessary
    ---------------------------------------
    ---------------------------------------

    // Rotate if there is an obstacle
    ---------------------------------------
    ---------------------------------------

    // Need to find some balls
    ---------------------------------------
    ---------------------------------------

    // Somehow pick up a ball
    ---------------------------------------
    ---------------------------------------

    // What if there is more than one ball?
    ---------------------------------------
    ---------------------------------------

  }
}
```

The Wrong Way – An Example

```
void moveForward( int time ) {

  while ( t < time ) {

    // Drive forward a bit
    ---------------------------------------
    ---------------------------------------

    // Check ir sensor and stop if necessary
    ---------------------------------------
    ---------------------------------------

    // Rotate if there is an obstacle
    ---------------------------------------
    ---------------------------------------

    // Need to find some balls
    ---------------------------------------
    ---------------------------------------

    // Somehow pick up a ball
    ---------------------------------------
    ---------------------------------------

    // What if there is more than one ball?
    ---------------------------------------
    ---------------------------------------

  . . .
```

```
    . . .

    // Need to find some goals
    ---------------------------------------
    ---------------------------------------

    // What if there are no goals visible?
    ---------------------------------------
    ---------------------------------------

    // Drop off some balls
    ---------------------------------------
    ---------------------------------------

    // Find more balls I guess
    ---------------------------------------
    ---------------------------------------

    // Make sure to ignore balls in goal
    ---------------------------------------
    ---------------------------------------

    // Try to go somewhere new
    ---------------------------------------
    ---------------------------------------

  }
}
```

The Wrong Way

- Monolithic code
  - Monster to maintain
  - Hard to debug
  - Confusing
- There's a better way…

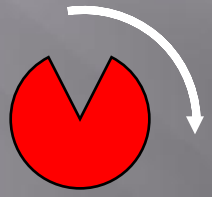
Overview

- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

Behaviors

- Defined as: a set or series of acts regarded as a unified whole
- Should be simple, well-defined, self-contained, and independently testable

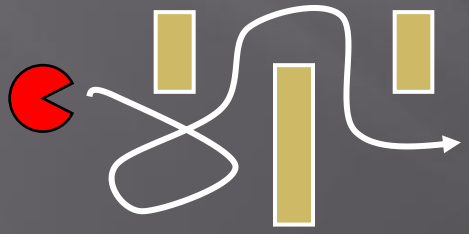
Behaviors



Turn right 90°

Go forward until reach obstacle

Capture a ball



Explore playing field

Complex Behaviors

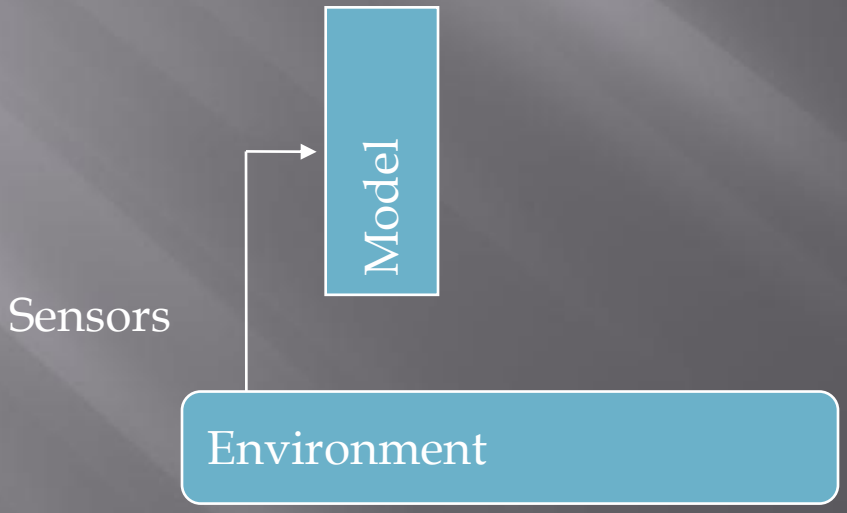
- Composed of primitive behaviors acting as building blocks.

Overview

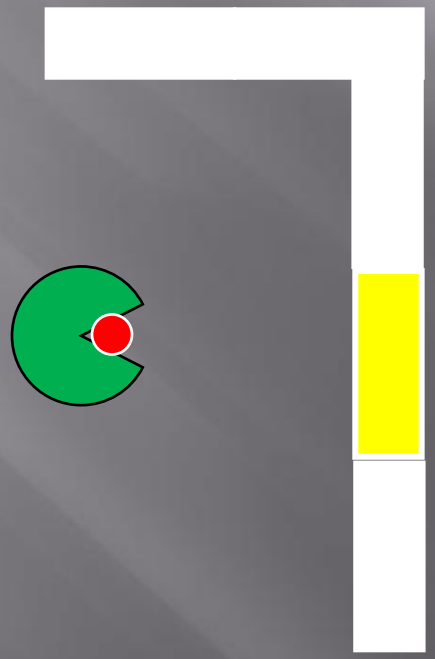
- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

Model Plan Act

- Use sensors to create a model of the world.



Model – Outside Perspective

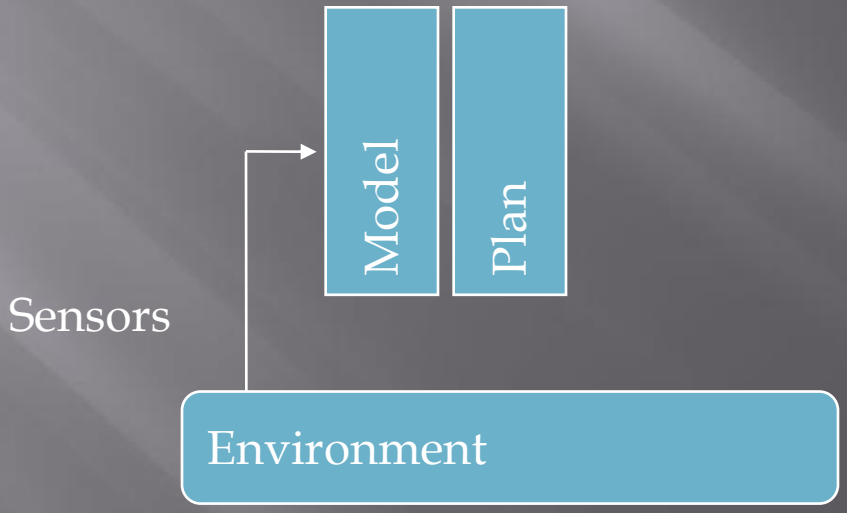


Model – Robot Perspective

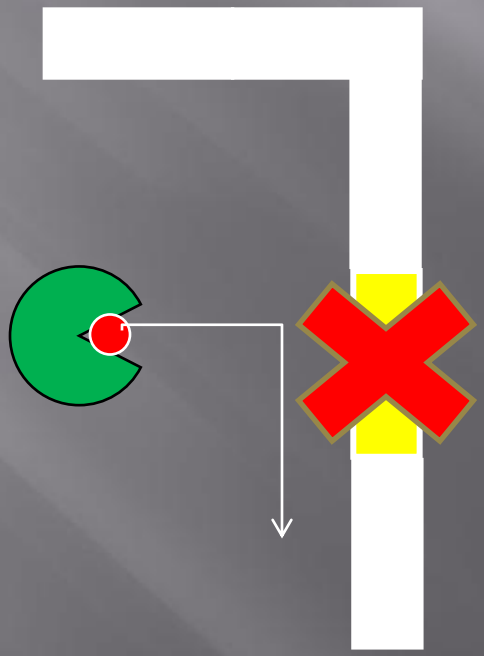
- I have 1 red ball.
- I see a yellow goal in front of me.
- I see a wall 30 cm to my left.
- I see open space to my right.
- I have traveled 3 ft since my last observation
- …
- …
- …

Model Plan Act

- Use sensors to create a model of the world
- Create a plan



Plan – Outside Perspective

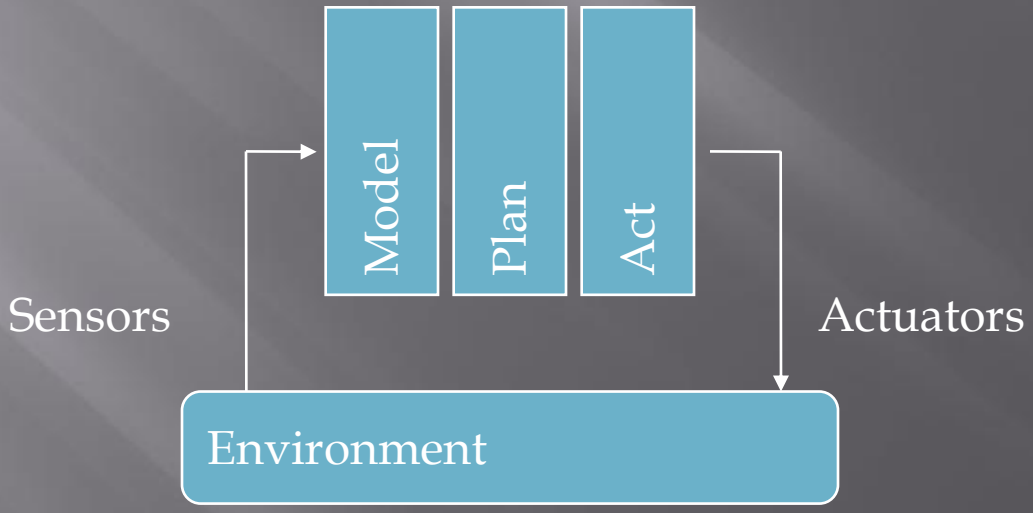


Plan – Robot Perspective

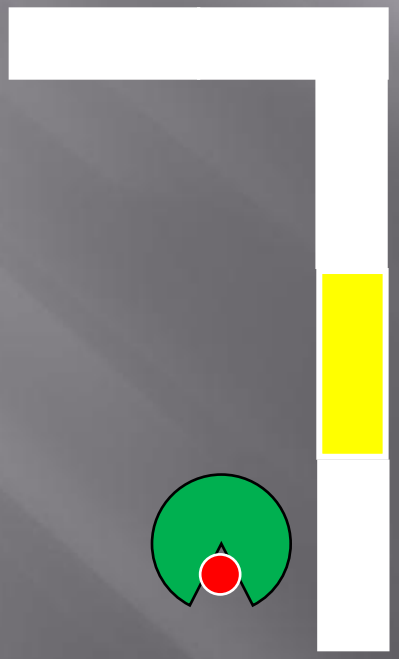
- I don't want to dump my ball, I'll get negative points
- Can't go left
- I just came from behind me
- Wall follow the wall in front of me!

Model Plan Act

- Use sensors to create a model of the world
- Create a plan
- Act on it by stringing together behaviors



Act – Outside Perspective

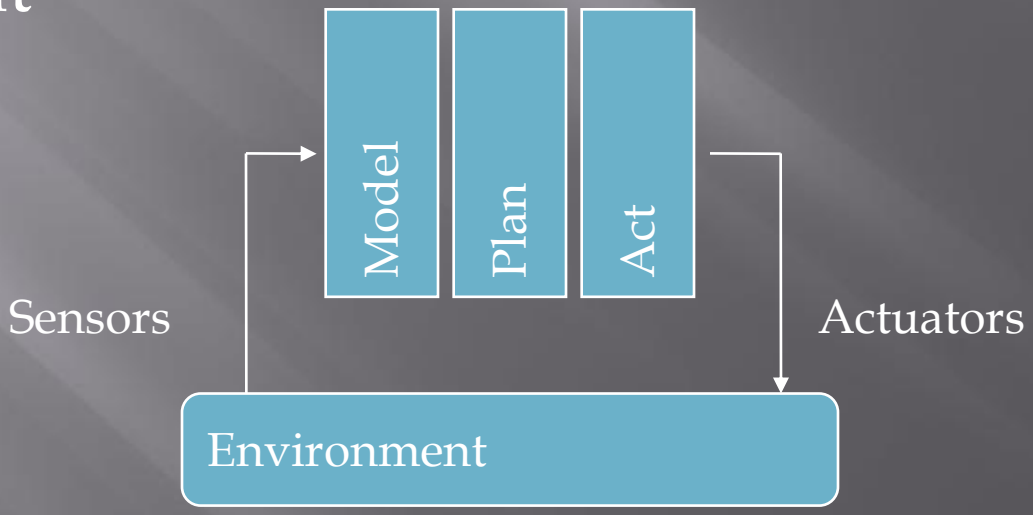


Act – Robot Perspective

- Go forward until IR reports wall is 15 cm away
- Turn 90° right
- Wall follow

Model Plan Act

- Use sensors to create a model of the world
- Create a plan
- Act on it by stringing together behaviors
- Repeat



Advantages and Disadvantages

- Advantages
  - Global knowledge in the model enables optimization
  - Can make provable guarantees about the plan

Advantages and Disadvantages

- Advantages
  - Global knowledge in the model enables optimization
  - Can make provable guarantees about the plan
- Disadvantages
  - Must implement all functional units before any testing
  - Computationally intensive
  - Requires very good sensor data for accurate models
  - Models are inherently an approximation
  - Works poorly in dynamic environments

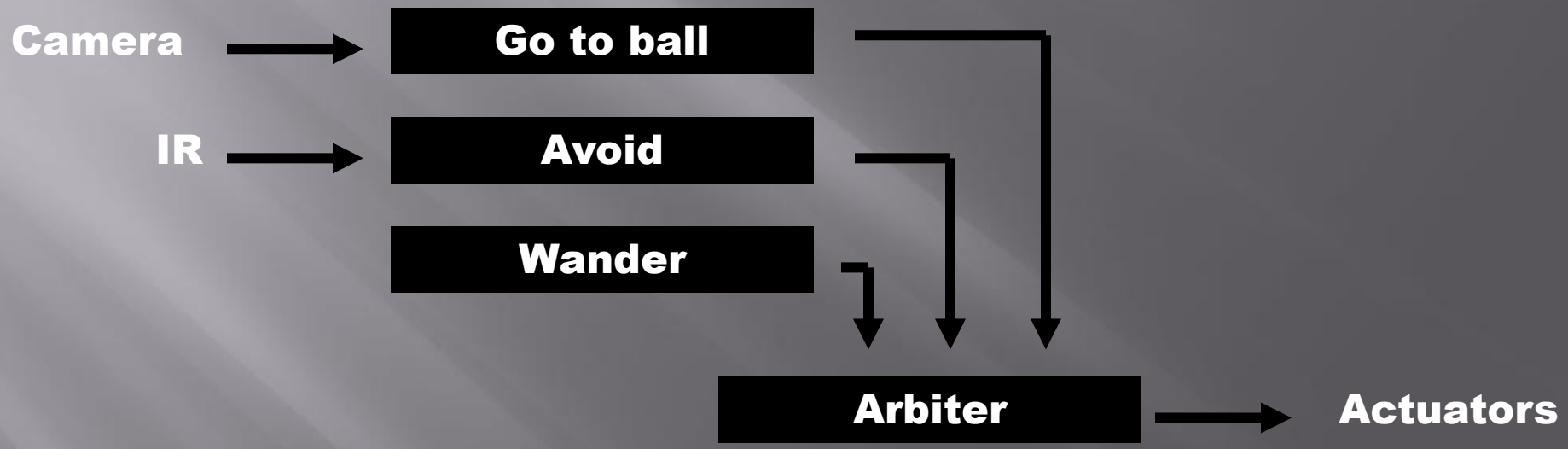
Overview

- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

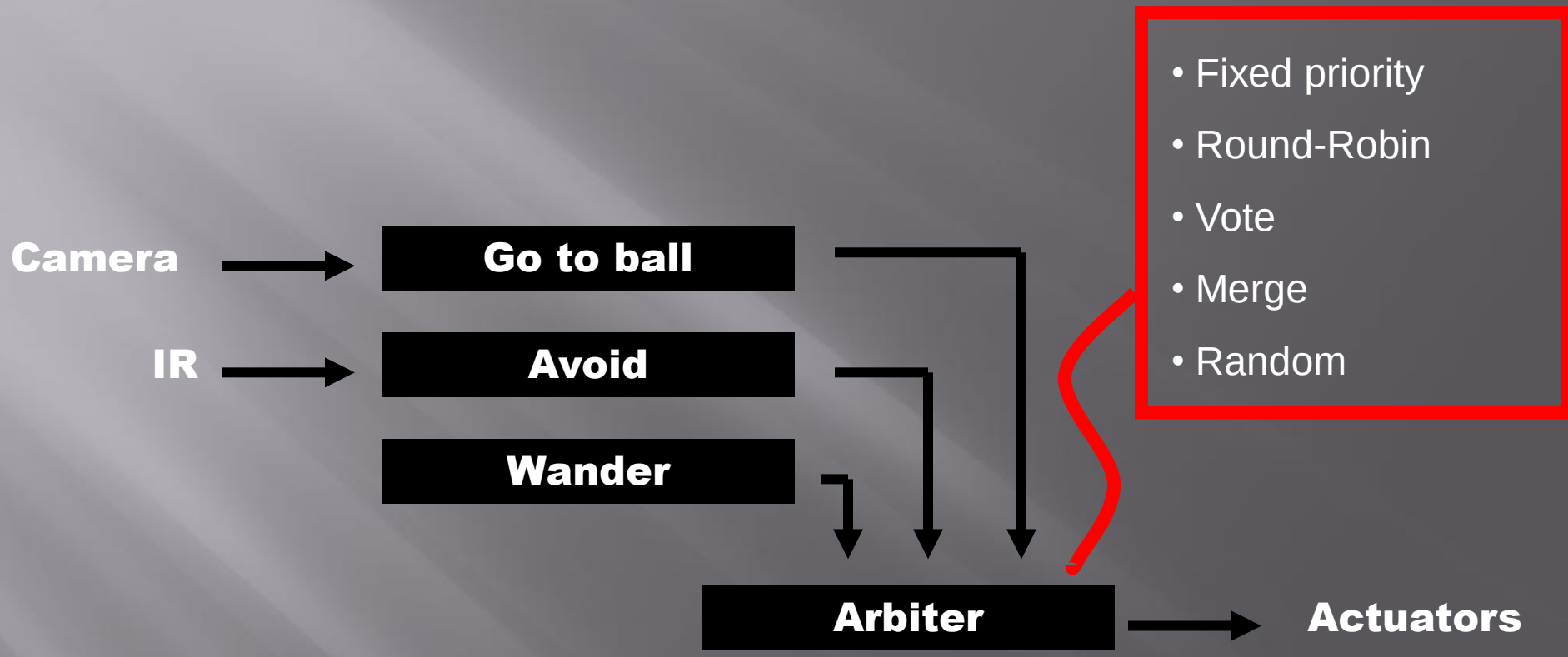
Emergent Behavior

- Sensors feed data into behaviors
- Behaviors decide what to do based on given sensor input
- Arbiter decides which behavior gets priority
- Arbiter passes command to actuators

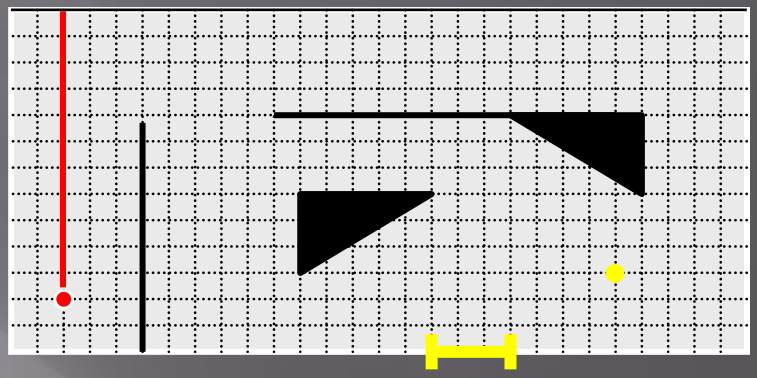
Emergent Behavior



Emergent Behavior

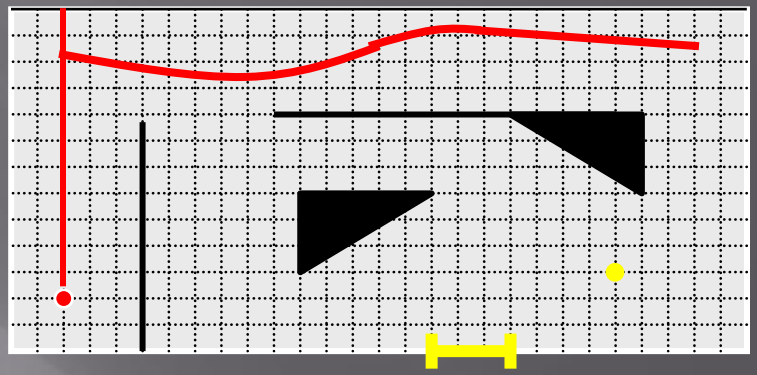


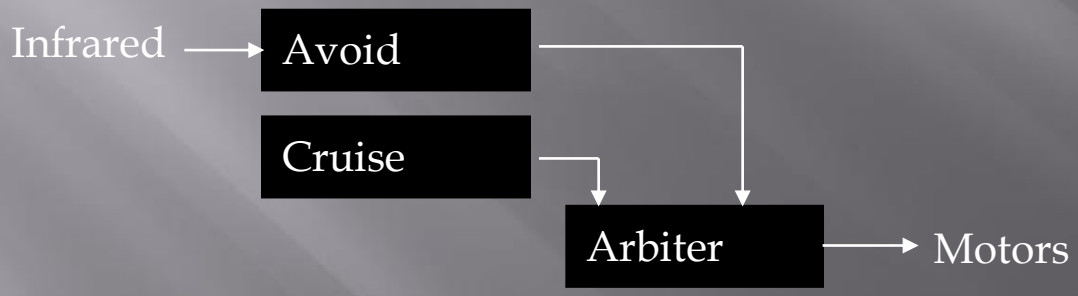
Emergent Behavior - Example



Cruise behavior simply moves robot forward

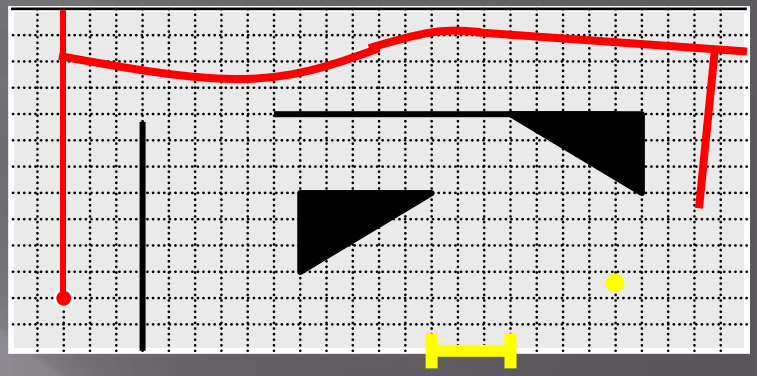
Emergent Behavior - Example

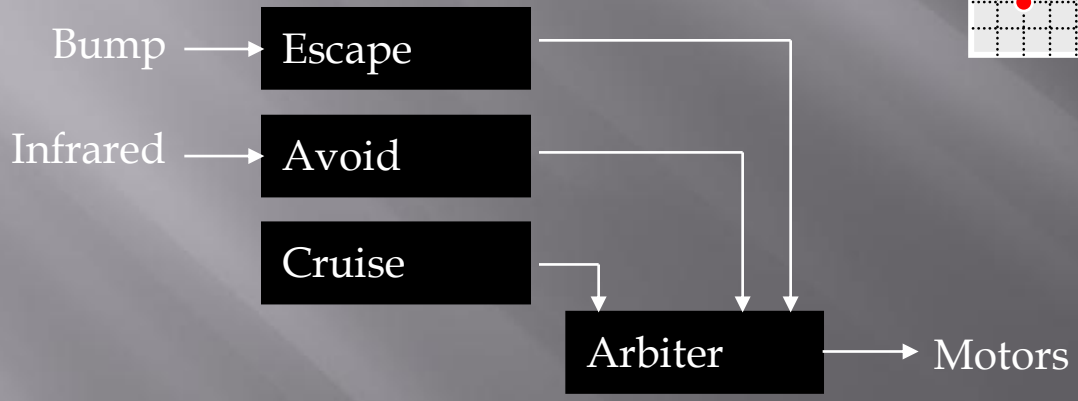




Left motor speed inversely proportional to left IR range
Right motor speed inversely proportional to right IR range
If both IR < threshold stop and turn right 120 degrees

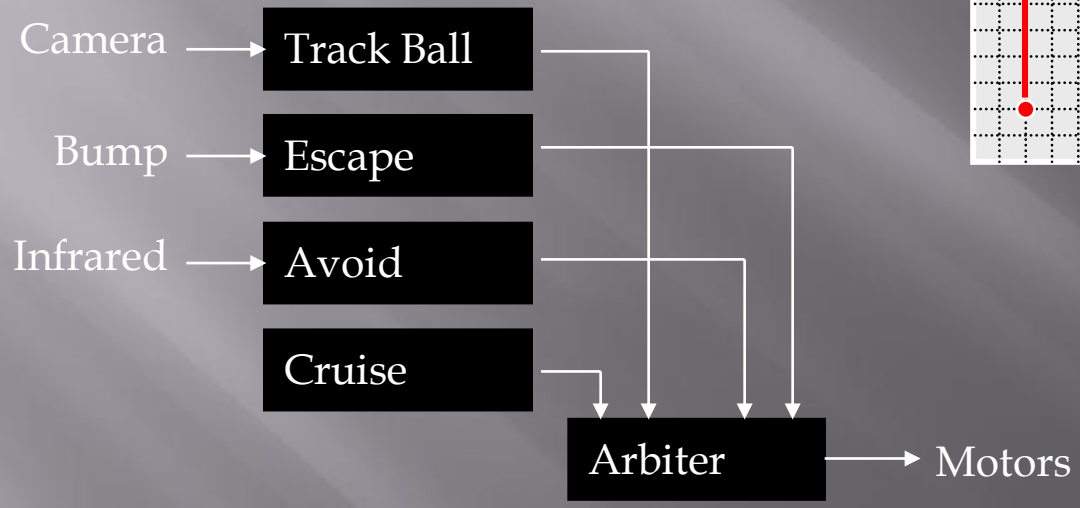
Emergent Behavior - Example

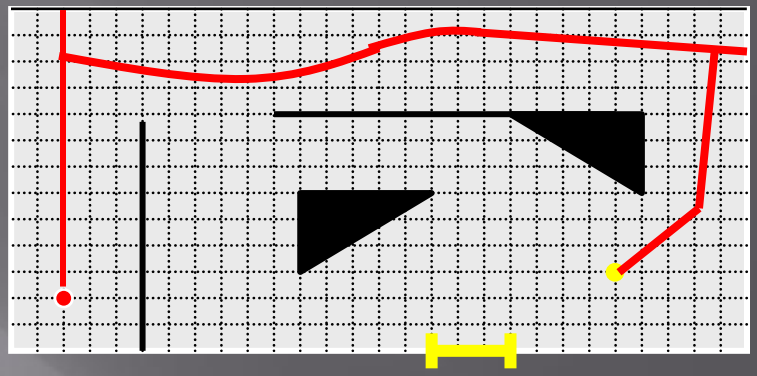




Escape behavior stops motors,
backs up a few inches, and turns right 90 degrees

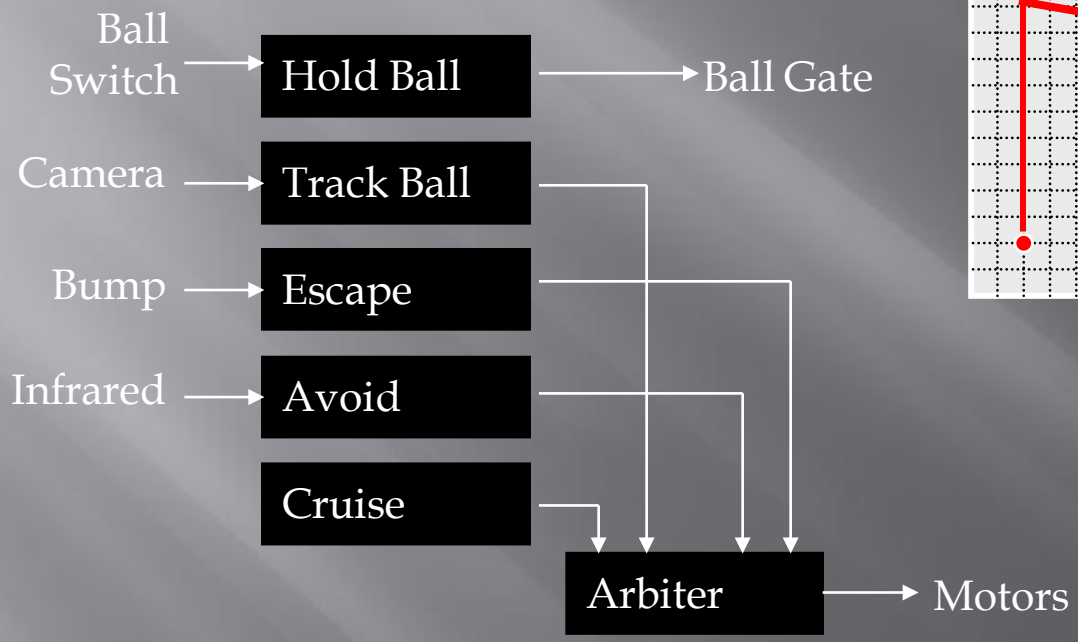
Emergent Behavior - Example

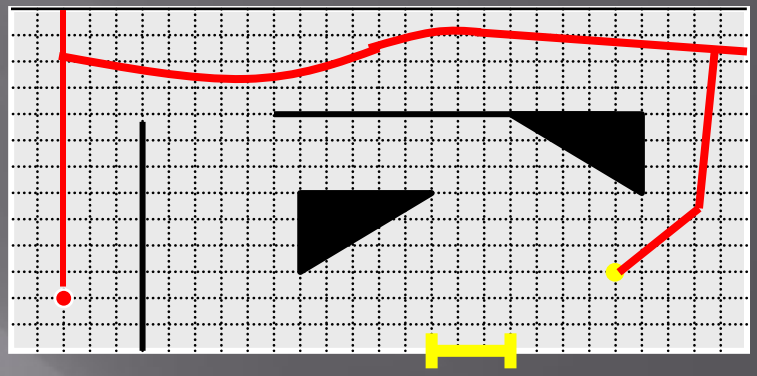




The track ball behavior adjusts the
motor differential to steer the robot towards the ball

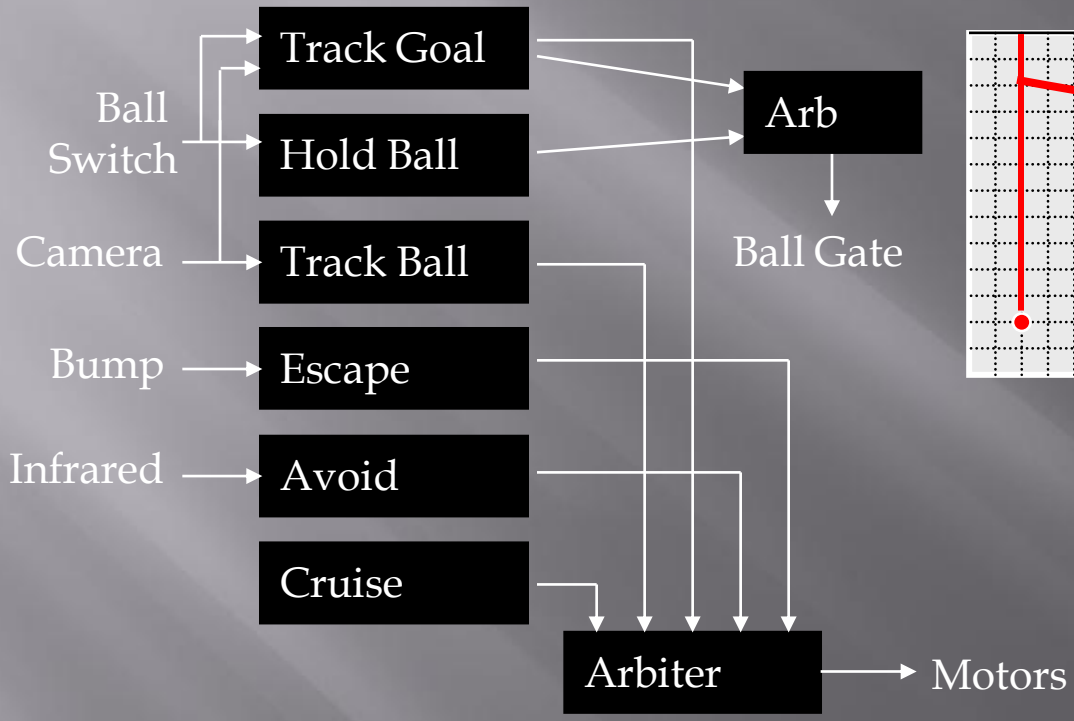
Emergent Behavior - Example

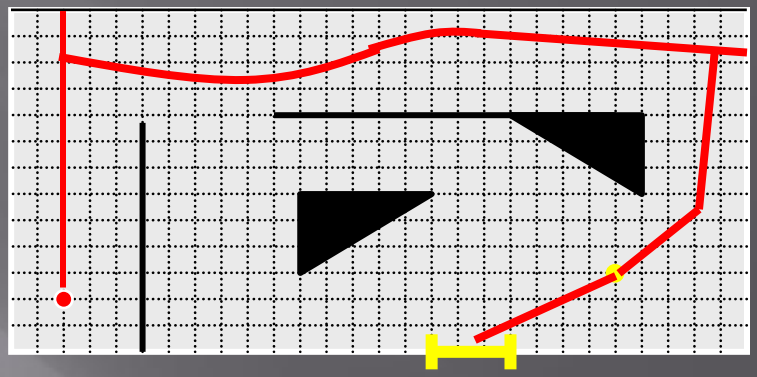




Hold ball behavior simply closes ball gate when ball switch is depressed

Emergent Behavior - Example





The track goal behavior opens the ball gate and adjusts the motor differential to steer the robot towards the goal

Advantages and Disadvantages

- Advantages
  - Incremental development is very natural
  - Modularity makes experimentation easier
  - Cleanly handles dynamic environments
  - Arbiter can easily shut down robot

Advantages and Disadvantages

- Advantages
  - Incremental development is very natural
  - Modularity makes experimentation easier
  - Cleanly handles dynamic environments
  - Arbiter can easily shut down robot
- Disadvantages
  - Not predictable
  - No performance or completeness guarantees
  - Debugging can be very difficult

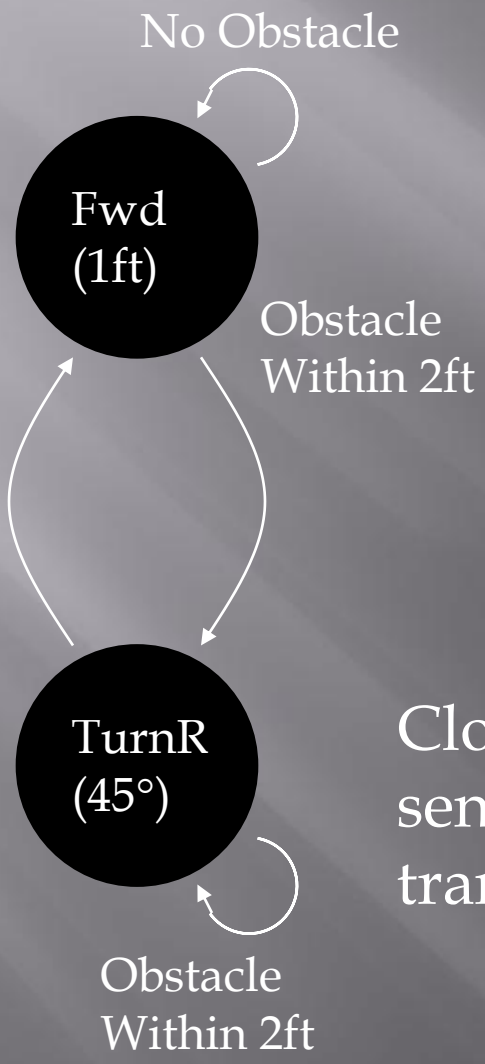
Overview

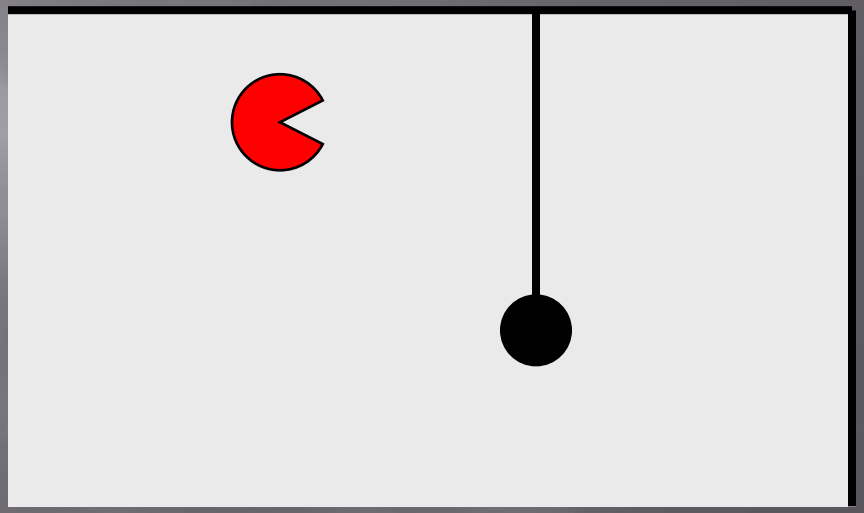
- The Big Picture
  - Your Goal
- The Wrong Way
- The Right Way
  - Model Plan Act
  - Emergent Behavior
  - Finite State Machines

Finite State Machines

- A state is an instance of a behavior
- A state can only transition into certain other states
- Combines static and dynamic behavior

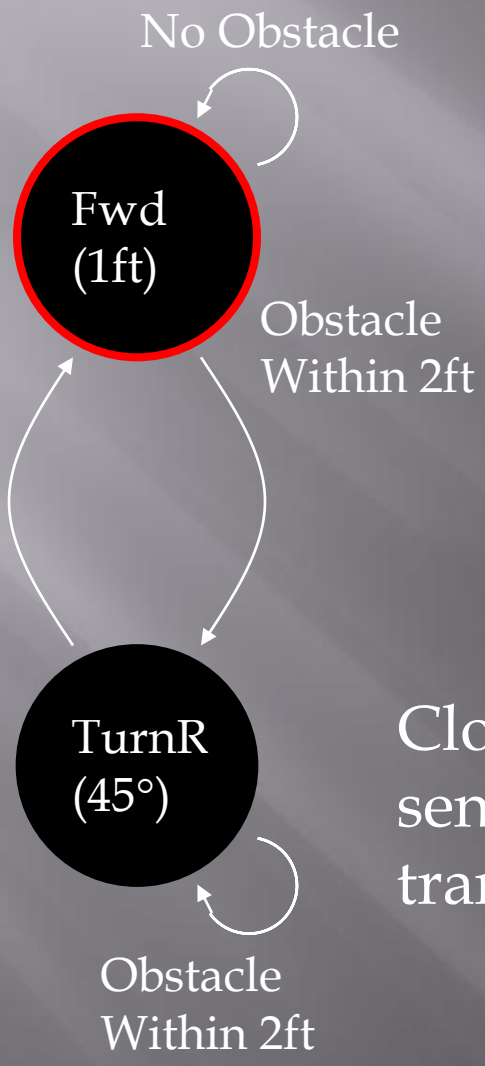
Simple Example

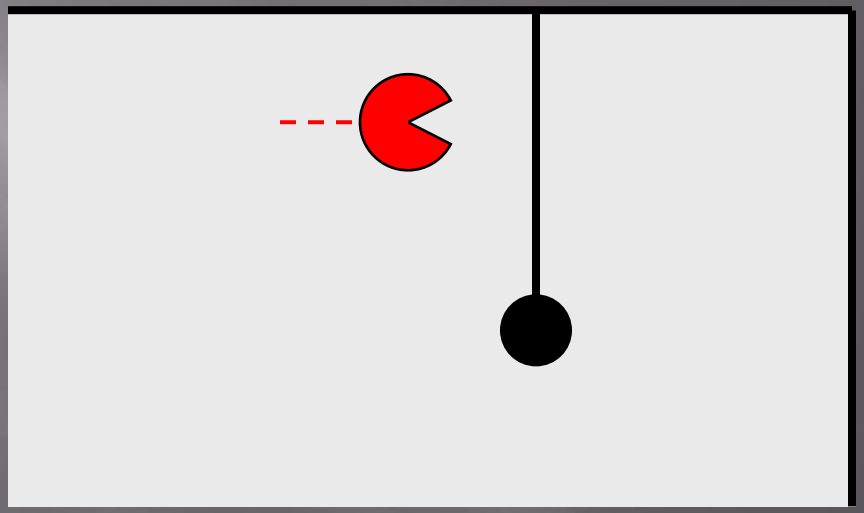




Closed loop finite state machines use sensor data as feedback to make state transitions

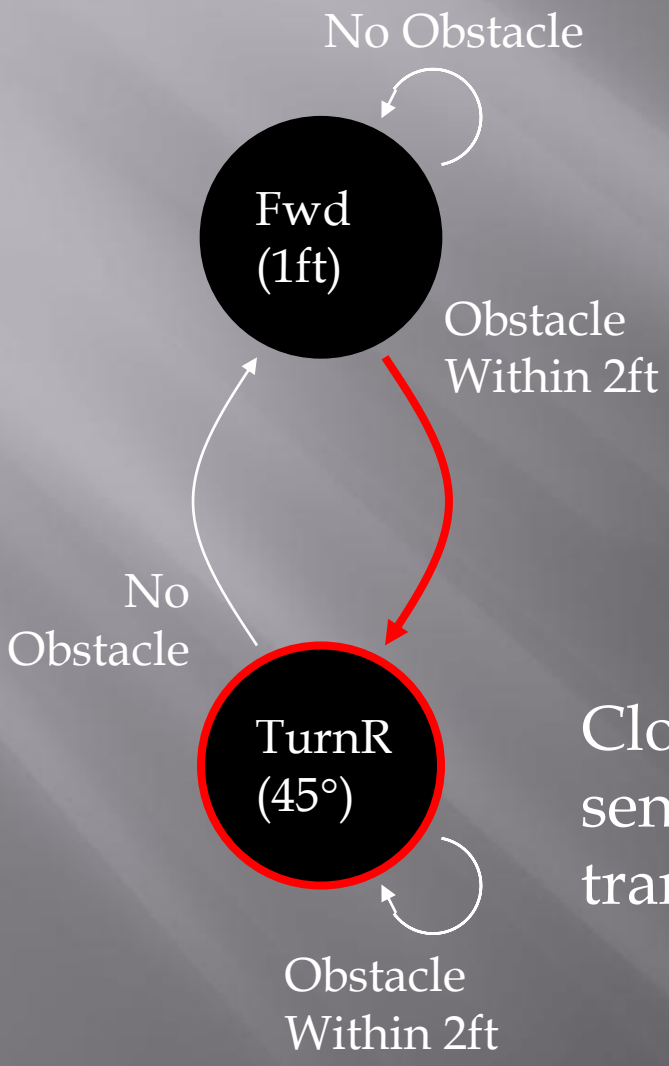
Simple Example

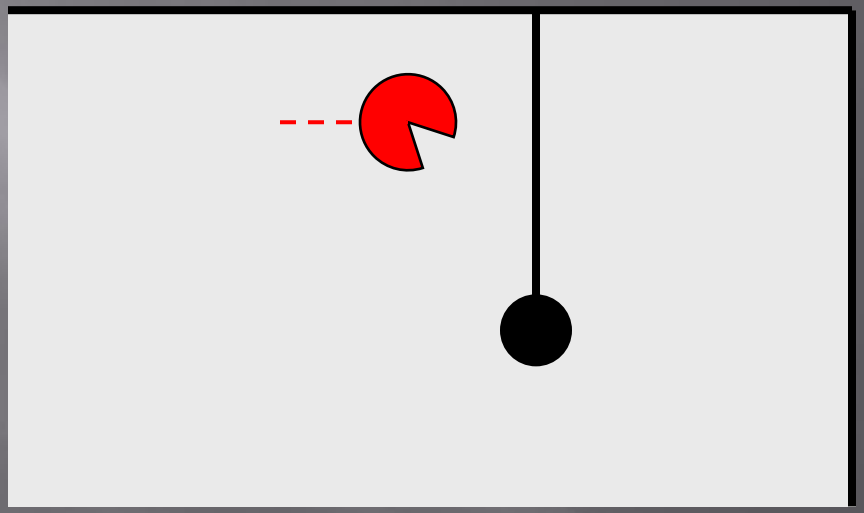




Closed loop finite state machines use sensor data as feedback to make state transitions

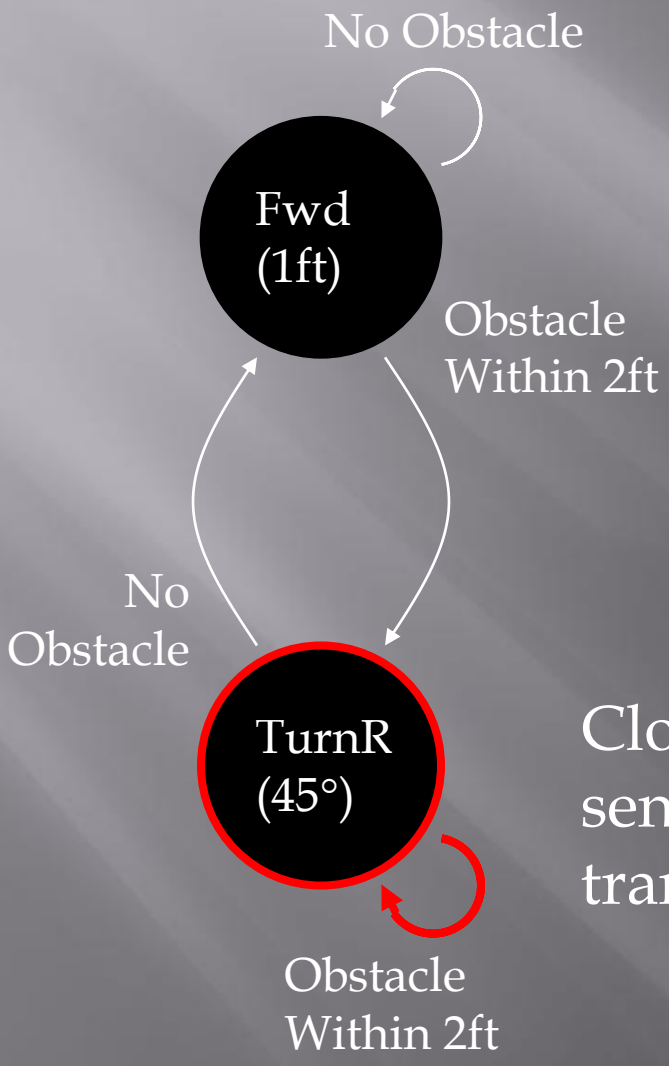
Simple Example

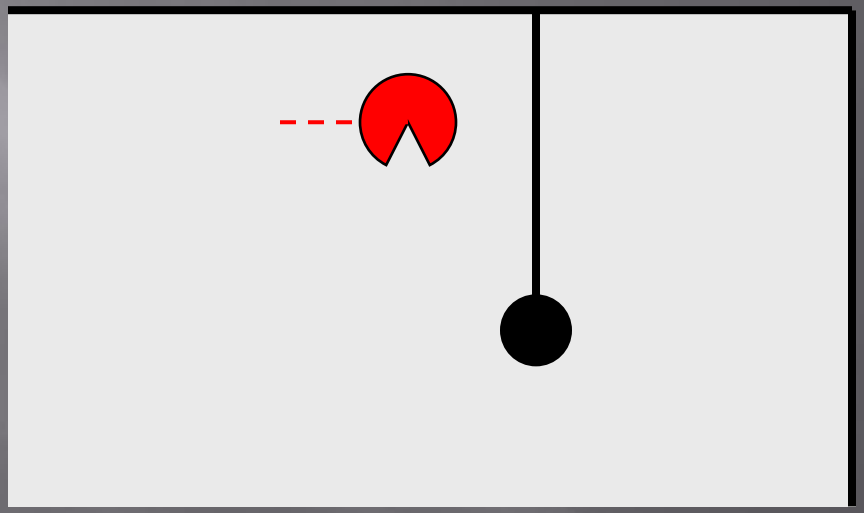




Closed loop finite state machines use sensor data as feedback to make state transitions

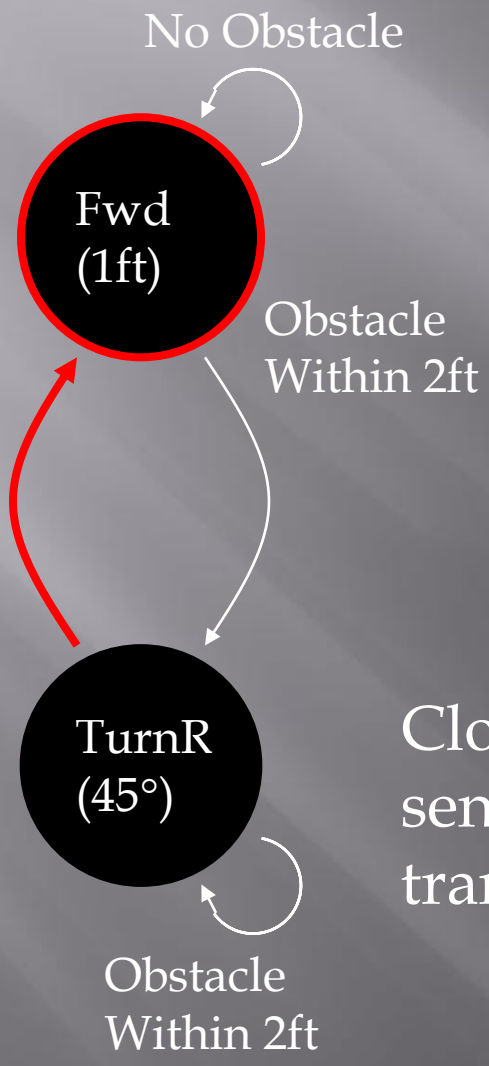
Simple Example

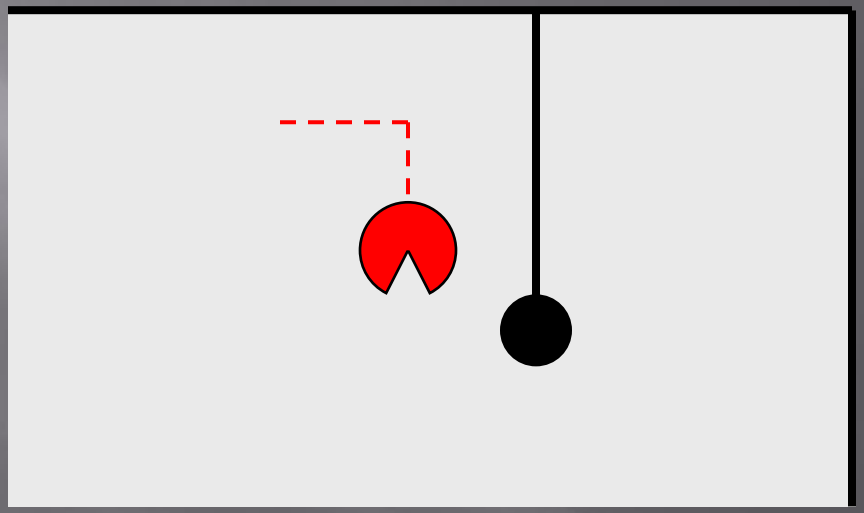




Closed loop finite state machines use sensor data as feedback to make state transitions

Simple Example





Closed loop finite state machines use sensor data as feedback to make state transitions

Implementing in Java

- Can use an enum to hold the value of the state and then switch on it

```
switch ( state ) {

  case States.Fwd_1 :
    moveFoward(1);
    if ( distanceToObstacle() < 2 )
      state = TurnR_45;
    break;

  case States.TurnR_45 :
    turnRight(45);
    if ( distanceToObstacle() >= 2 )
      state = Fwd_1;
    break;

}
```

Implementing in Java Method 2

- Define a state interface

```
public interface IState{
  public IState performAction();
}
```

Implementing in Java Method 2

- Each state then becomes a class that implements the interface

```
public class Fwd implements
IState{

  private int dist;
  private IState nextState;
  public Fwd(int dist){
    this.dist = dist;
  }
  public IState performAction(){
    moveForward(dist);
    if (distanceToObstacle()<2){
      nextState = new TurnR(45d);
    }else{
      nextState = this;
    }
  return nextState;
}
```

```
public class TurnR implements
IState{

  private double angle;
  private IState nextState;
  public TurnR(double angle){
    this.angle = angle;
  }
  public IState performAction(){
    turnRight(angle);
    if (distanceToObstacle()>=2){
      nextState = new Fwd(1);
    }else{
      nextState = this;
    }
  return nextState;
}
```

Implementing in Java Method 2

- The states are run by a loop in the main thread

```
public static void Main(String[] args){

  IState currentState = new Fwd(1);
  while (time < 5min){
    currentState = currentState.performAction();
  }
}
```

Implementing in Java Method 2

- The states are run by a loop in the main thread

```
public static void Main(String[] args){

  IState currentState = new Fwd(1);
  while (time < 5min){
    currentState = currentState.performAction();
  }
}
```

- Can anyone see something wrong with this?

Implementing in Java Method 2

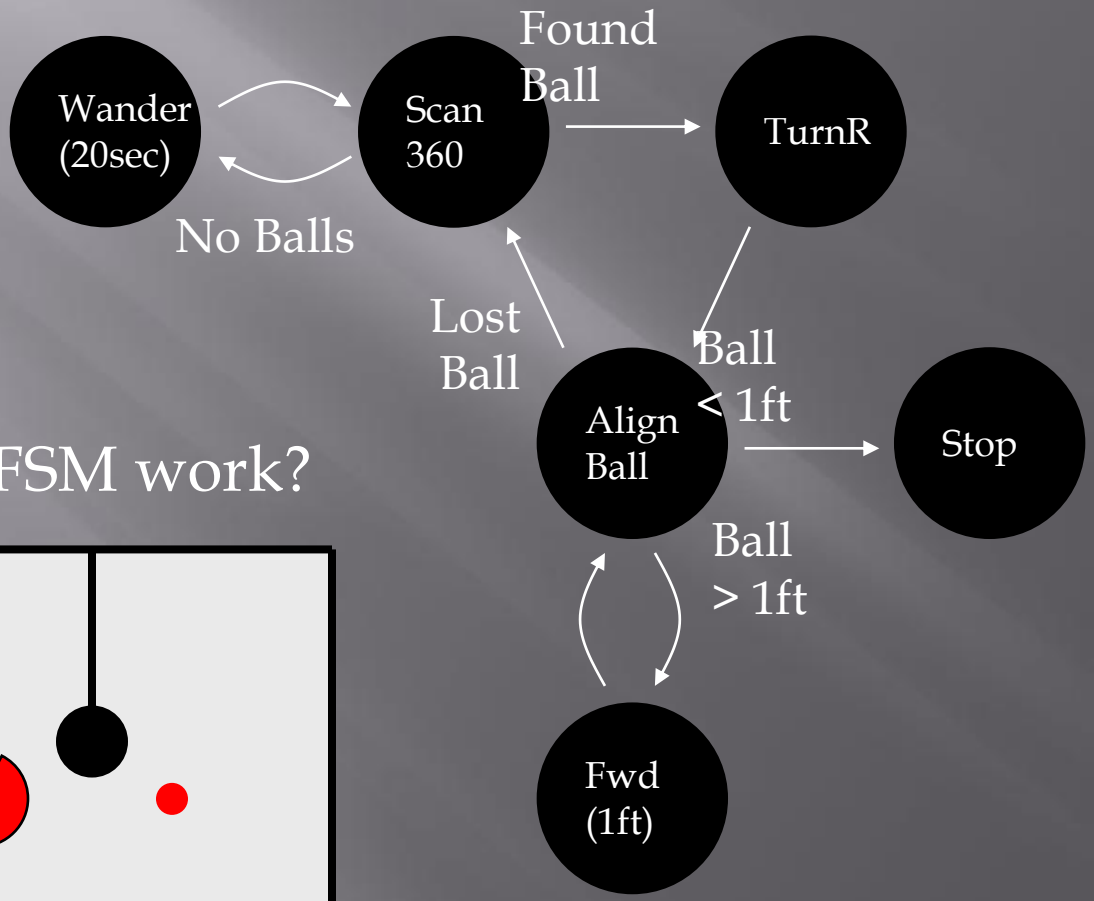
- The states are run by a loop in the main thread

```
public static void Main(String[] args){

  IState currentState = new Fwd(1);
  while (time < 5min){
    currentState = currentState.performAction();
  }
}
```

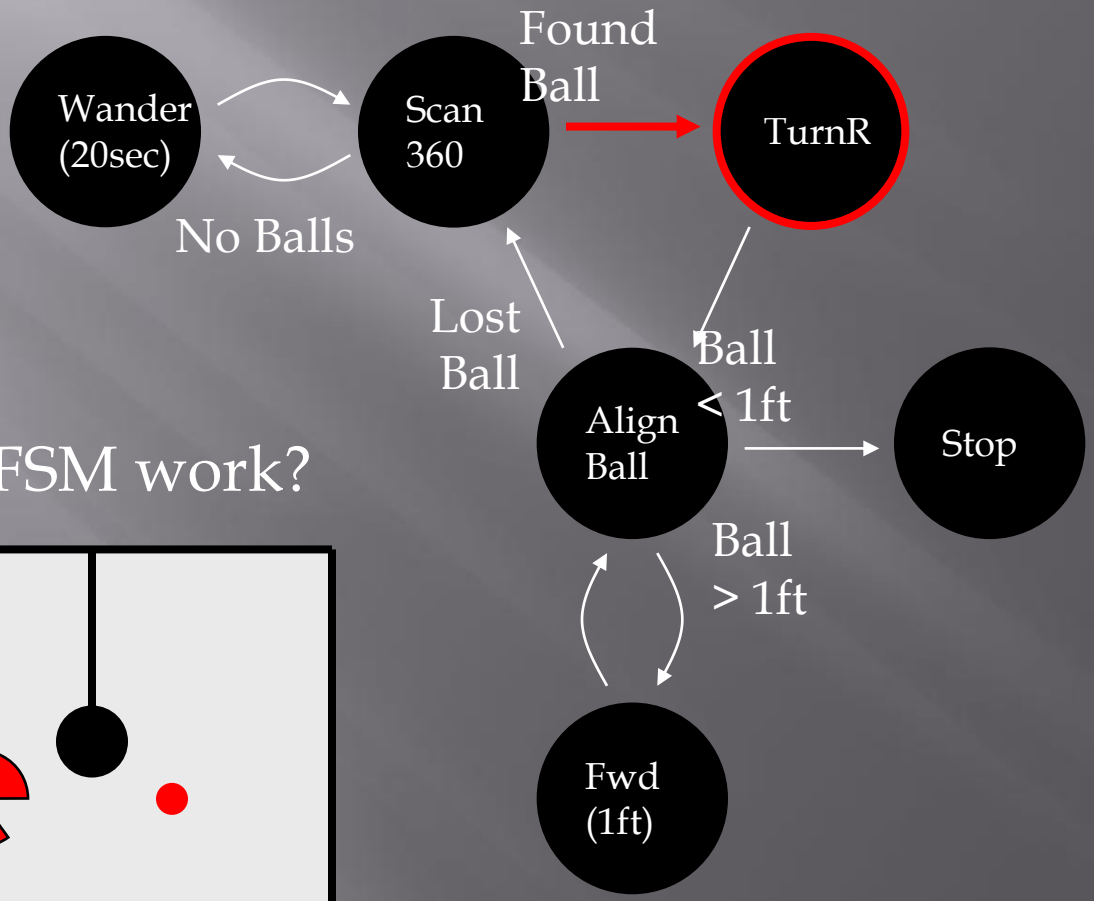
- Can anyone see something wrong with this?
- What if a state doesn't finish its task until 5:20 into the competition?

Complex Example



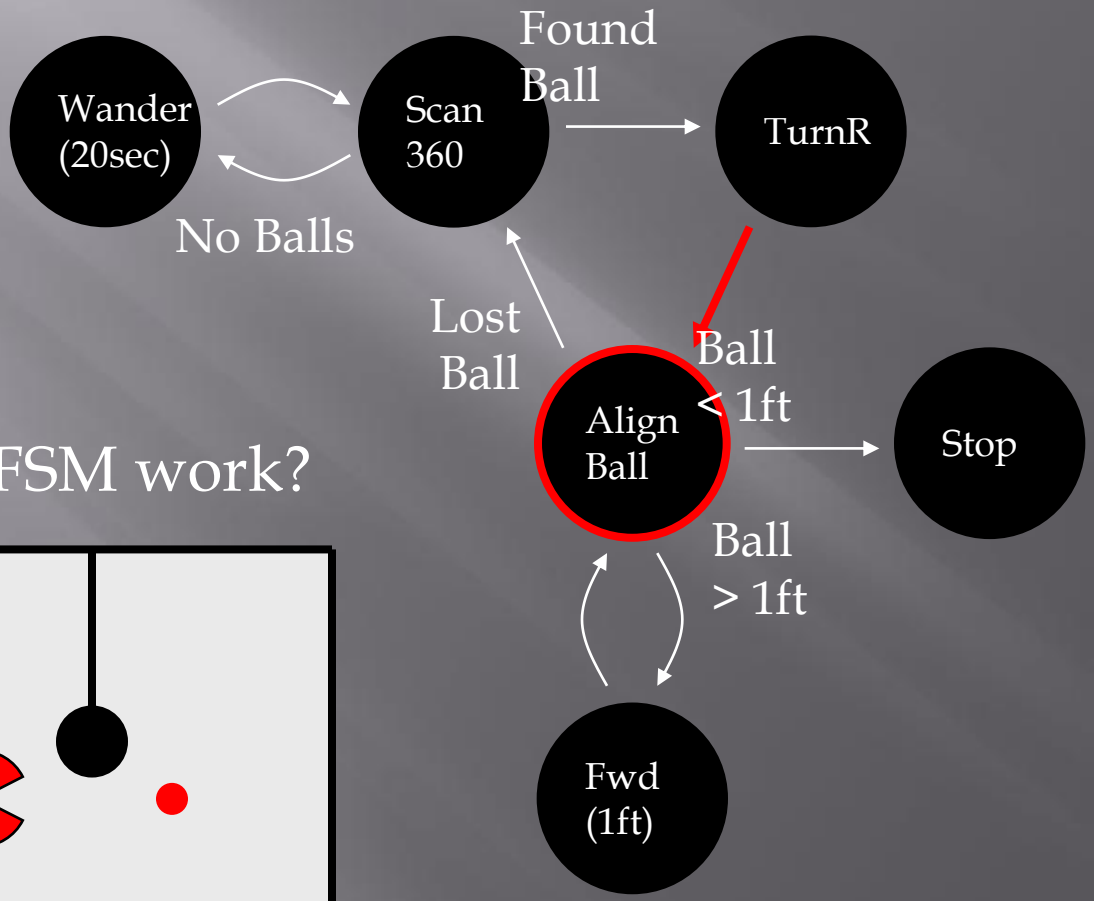
Does this FSM work?

Complex Example



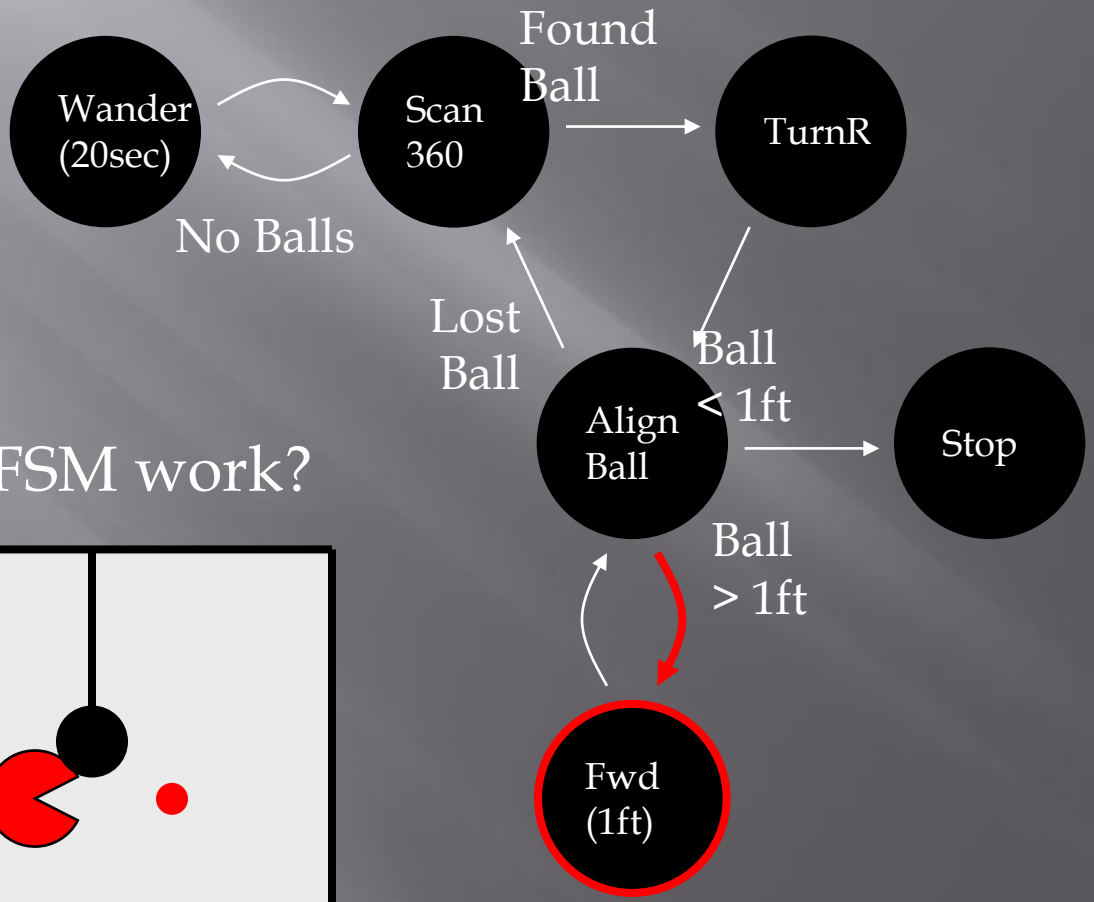
Does this FSM work?

Complex Example



Does this FSM work?

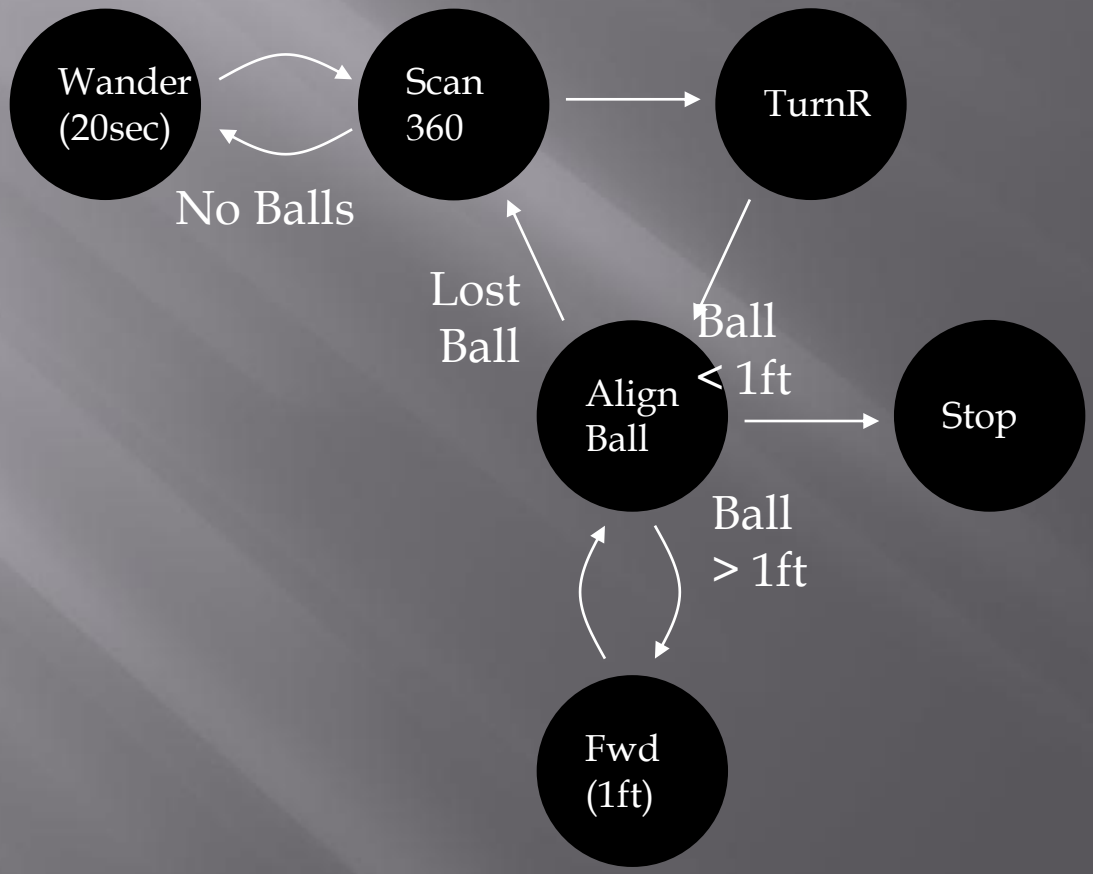
Complex Example



Does this FSM work?

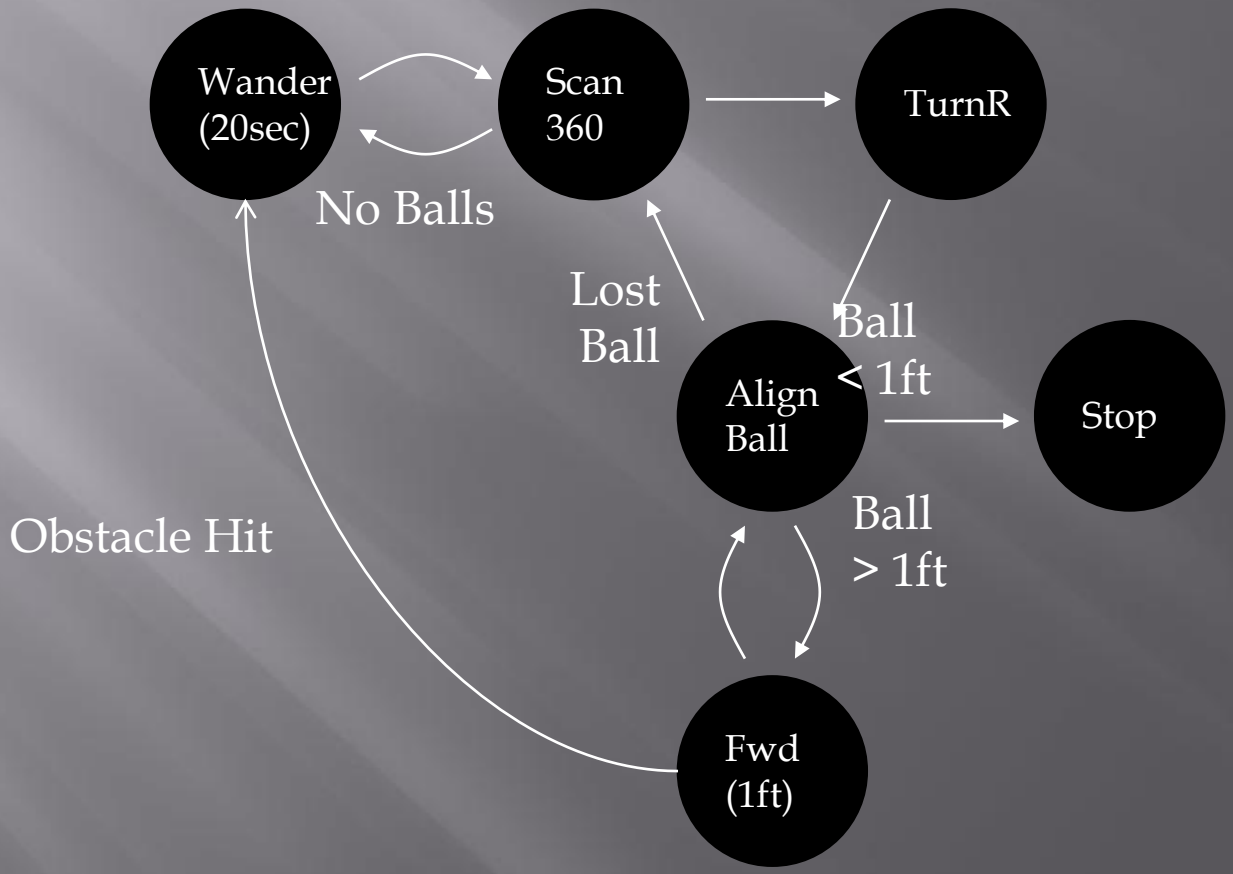
Complex Example

- It doesn't work. How do we improve it?



Complex Example

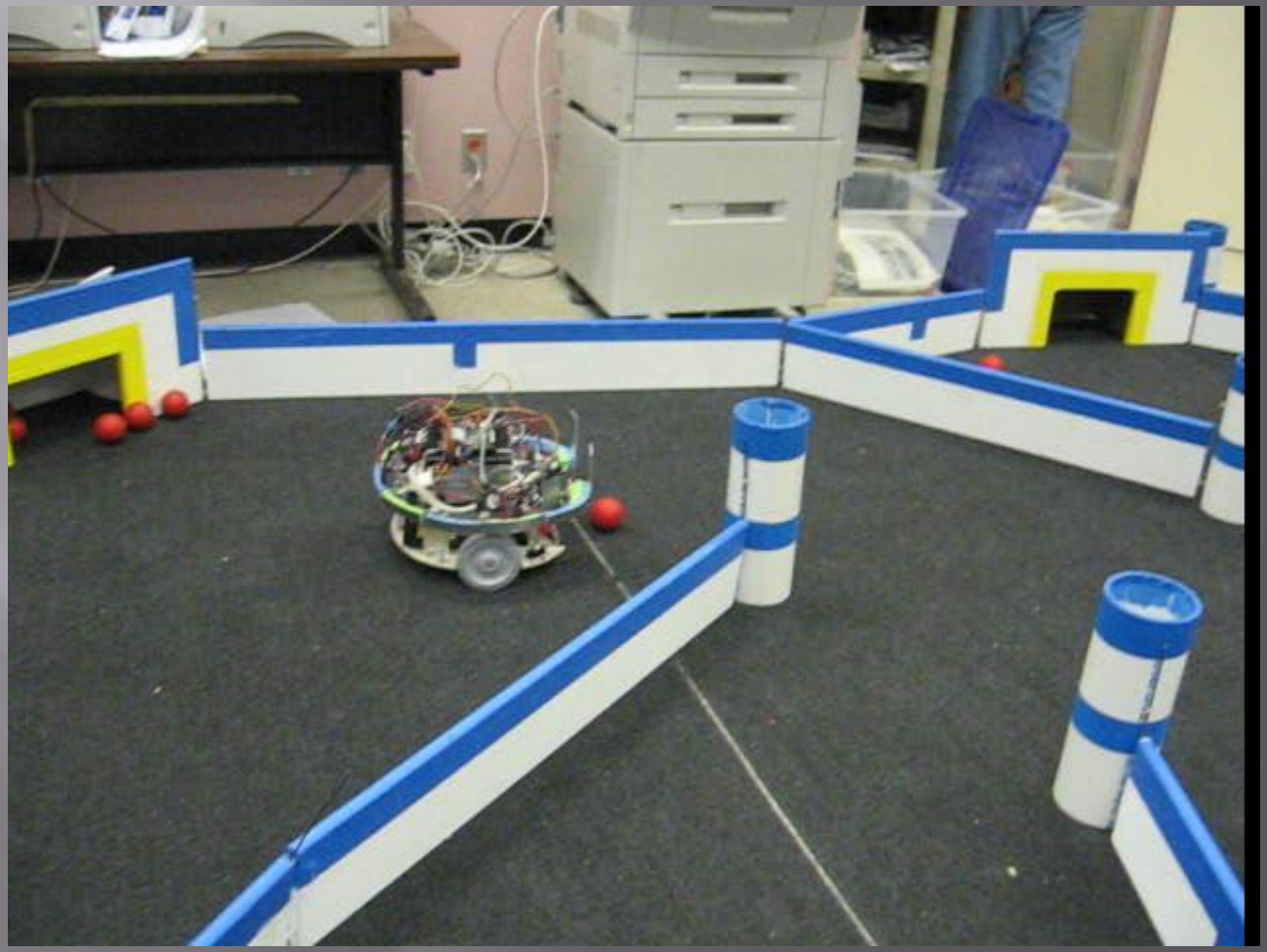
- It doesn't work. How do we improve it?



Advantages and Disadvantages

- Advantages
  - Combine Model Plan Act and Emergent Behavior
  - Predictable
  - Very Modular
  - Simple
- Disadvantages
  - Not as “free-flowing” as Emergent Behavior
  - Not as optimum as Model Act Plan

Video



Further Resources

- 6.005 Lecture Notes
  - http://stellar.mit.edu/S/course/6/fa07/6.005/
- Previous years' lecture notes

Thanks to:

- Valorie Morash
- Christopher Batten

Reminder

- Start thinking about Assignment 4 now!