Robots and People

Vlad Estivill-Castro (2016)

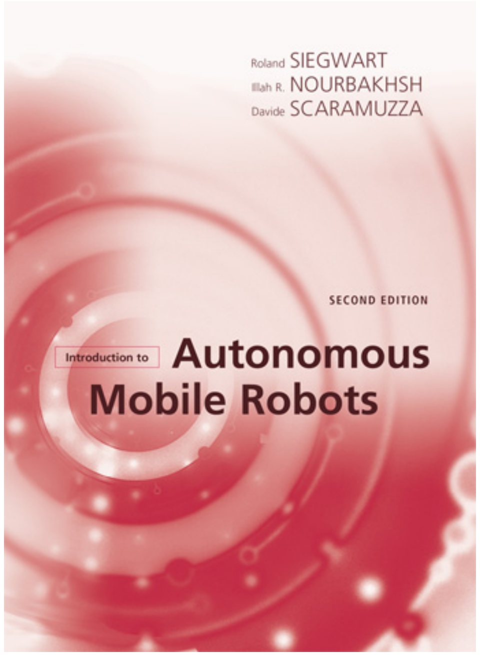
Robots for People

--- A project for intelligent integrated systems

Locomotion Concepts

Chapter 2 (textbook)

What is the course about?

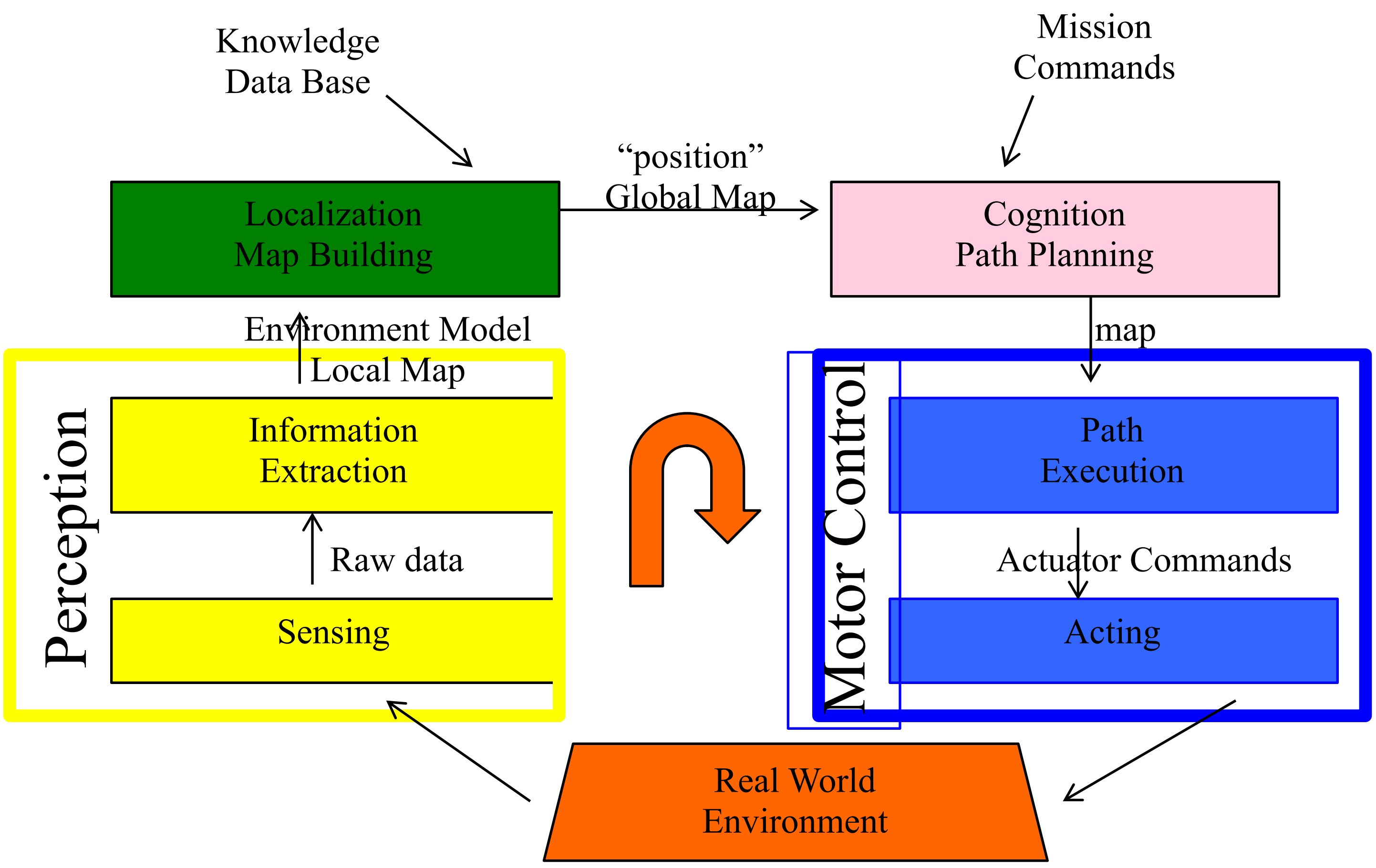


- textbook
- **Introduction to Autonomous Robots**
- second edition

  Roland Siegwart, Illah R. Nourbakhsh, and Davide Scaramuzza

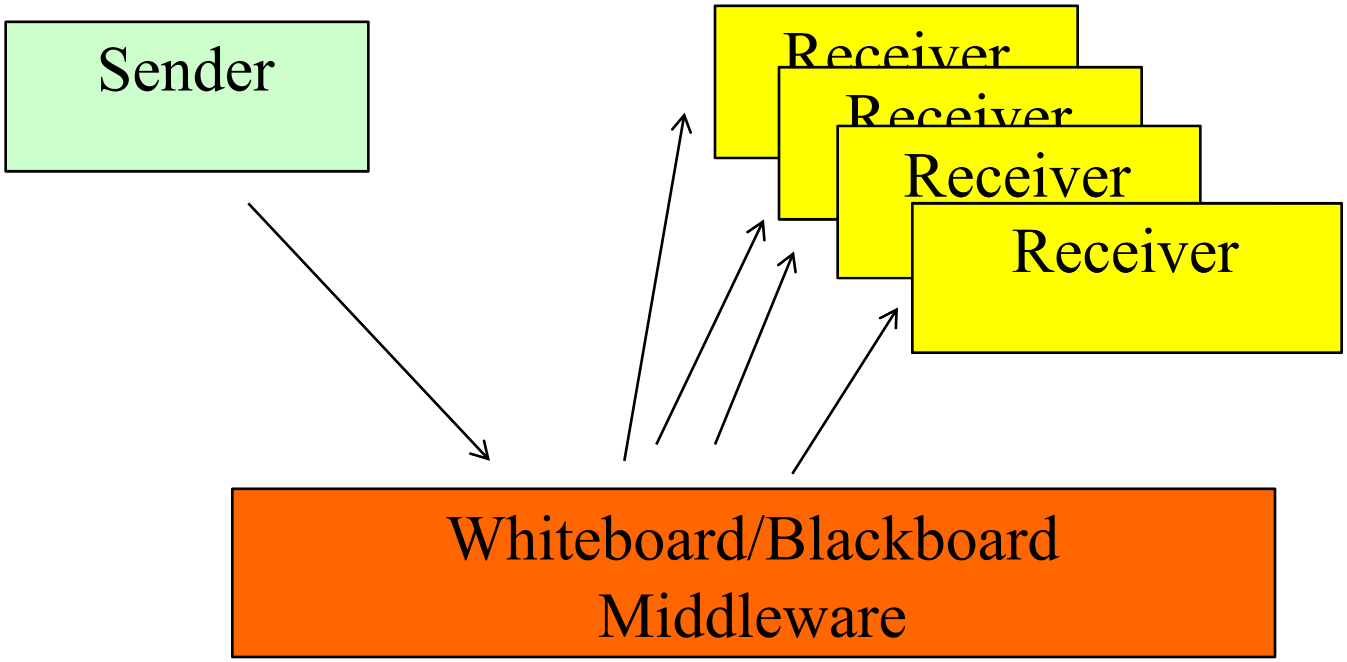
General Control Scheme for Mobile Robot Systems

- (Roland Siegwart, Illah Nourbakhsh, Davide Scaramuzza)



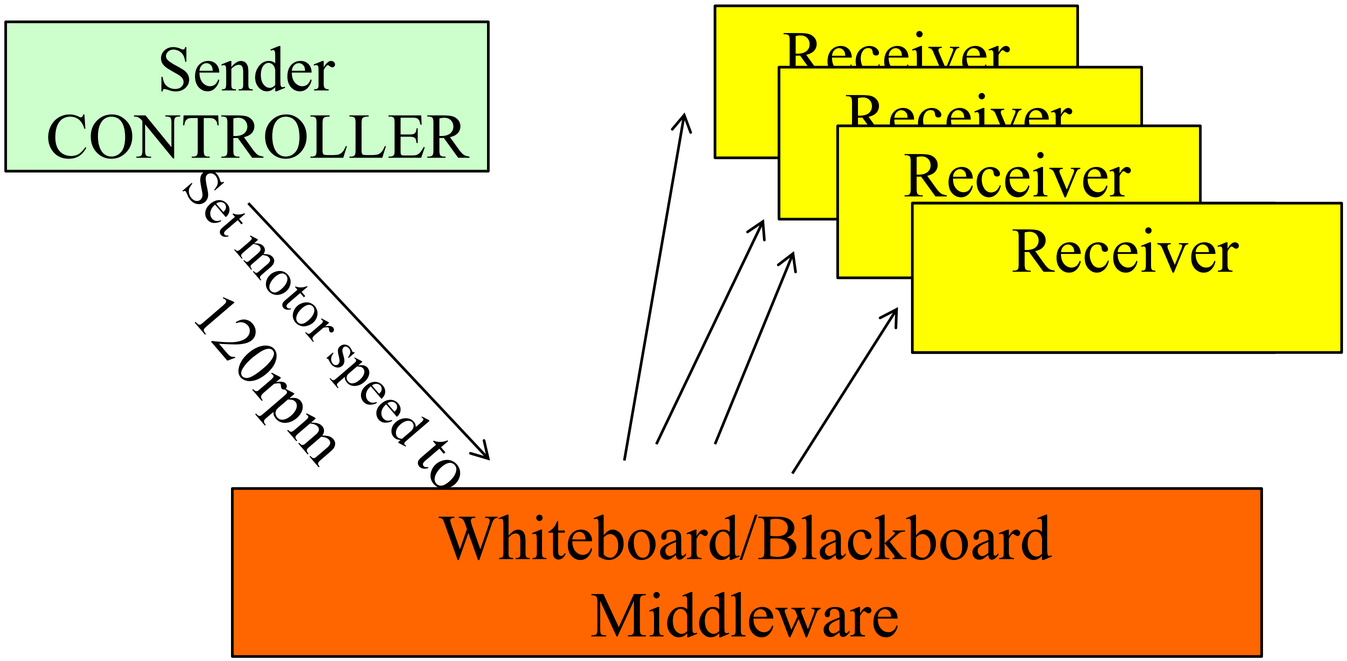
Blackboard architecture

- Reduce the number of APIs



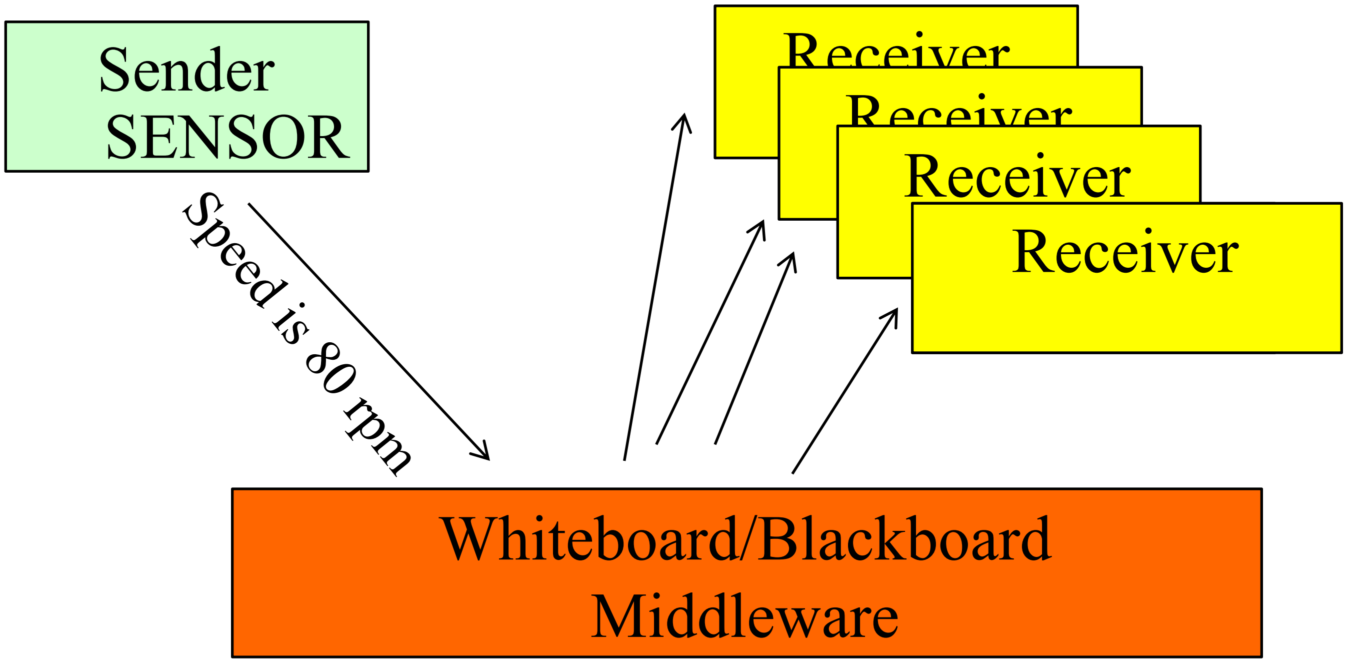
Blackboard architecture (example)

- Controller on an actuator



Blackboard architecture (example)

- Information producer

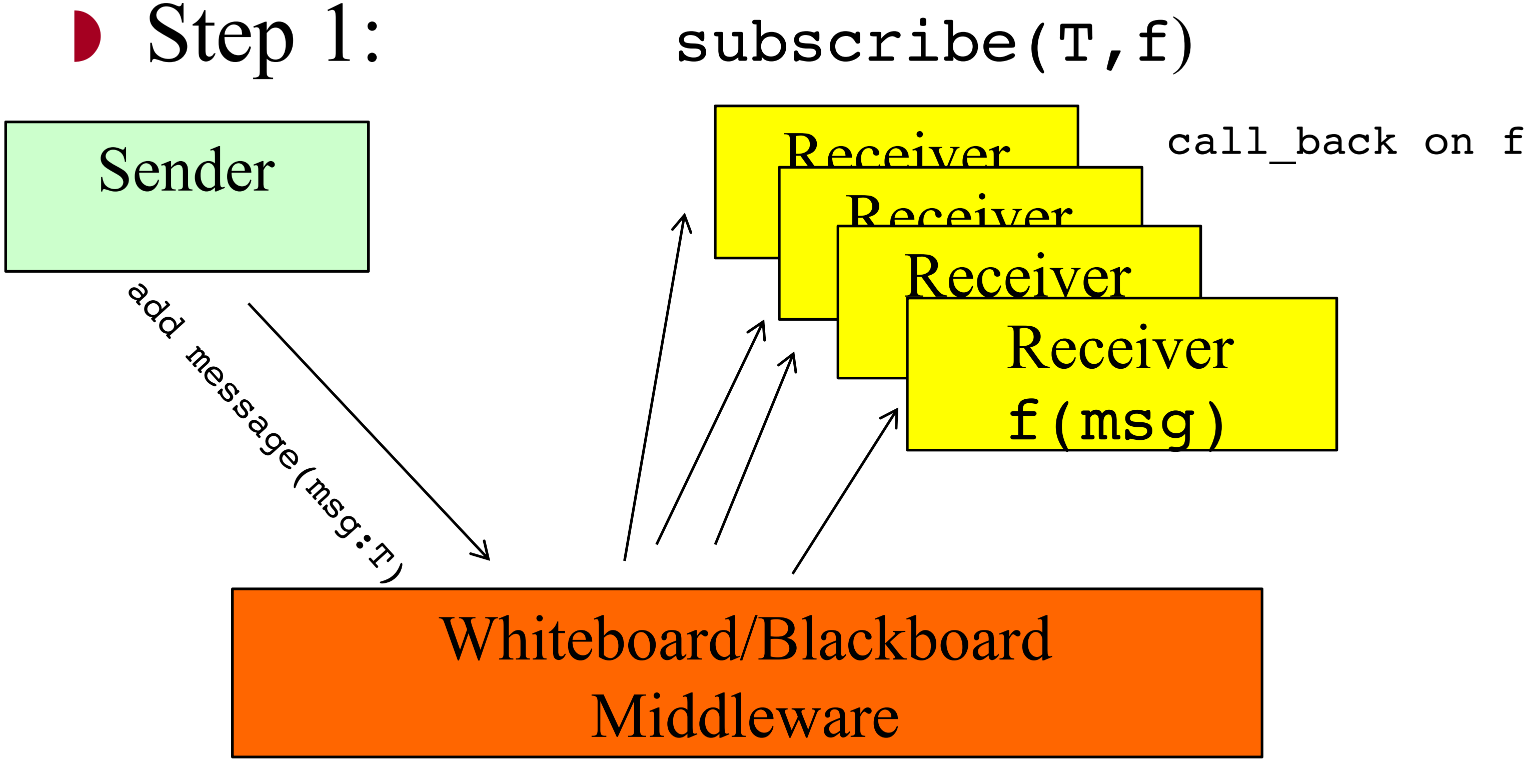


The main communication service

- Sender
  - `add message(msg:T)`
    - which is a non-blocking call
- Receiver
  - `subscribe(T,f)`: The receiver subscribes to messages of a certain type T (of an implied class) and essentially goes to sleep.
    - Subscription includes the name `f` of a function.
    - The blackboard will notify the receiver of the message `msg` every time someone posts for the given class T by invoking `f(msg)` (usually queued in a type `T` specific thread).
    - This is typically called **PUSH** technology.
  - `get_message(T)`: The receiver issues a get message to the blackboard that supplies the latest `msg` received so far for the type `T` .
    - This is usually called **PULL**.

*Blackboard architecture (**PUSH**)*

- Step 1:

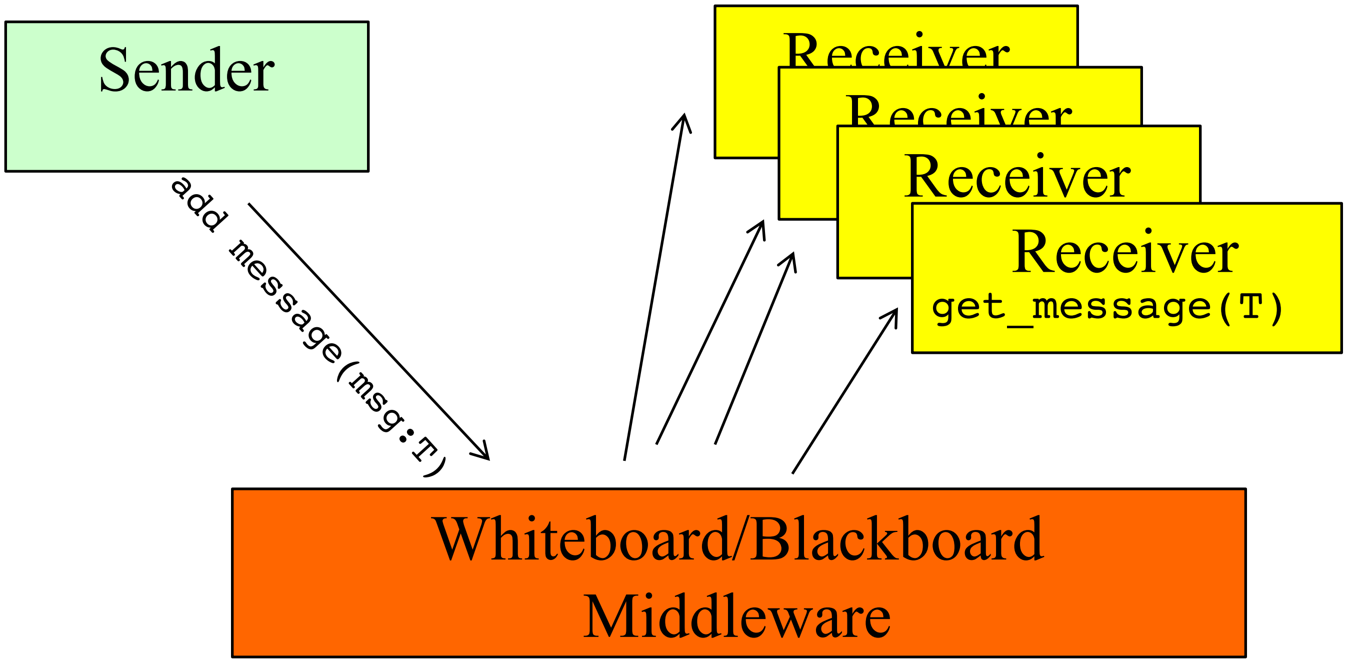


A popular trend and a critique

- **PUSH** technology
  - event driven
  - Assumptions
    - there will be sufficient computational resources to enact all the threads generated and to execute the subscribers' call-backs.
    - events would occur with enough sparsity that call-backs would be completed by the time they are executed again, or alternatively require to handle concurrency issues
  - coupling and timing consequences:
    - sub-scriber modules react or act on the data immediately, thus data is presumed to be as fresh and as recent as possible.
    - leads to more coupling
      - senders must eventually be slowed down or complex handshaking protocols have to be deployed in order to raise tolerance to message loss.

Blackboard architecture (PULL)

- Receivers work at their own pace



Robots and people

Example 1 (Sphere of control)

- Reporting/publishing a sensor value to the middleware
  - How to generate the proper hardware abstraction?
    - who is to say when messages are posted?
    - Is this defined by the subscriber, by the publisher, or by the API?
- The **PUSH** paradigm puts messages in the sphere of control of the publisher.
  - subscriber will need to keep up in order to not lose any messages, no matter whether that specific update rate is necessary or convenient for the receiver.
  - the timing of the subscriber is determined by (or **tightly coupled** with) the publisher.

Example 2 (Separation of Concerns)

- a controller that sends a message for an action to be initiated and then waits for that action to have started (or completed) before continuing (a ROS example)

```
ros::NodeHandle n;
ros::ServiceClient client = n.serviceClient
                              <speed_controller::Speed>("speed");
speed_controller::Speed srv;
srv.request.desiredSpeed = atoi(argv[1]); // set speed
if (client.call(srv))
  ROS_INFO("Speed: %d", (int)srv.response.reportedSpeed);
else
{
  ROS_ERROR("Failed to call service speed_controller");
  return EXIT_FAILURE;
}
```

THE CLIENT

Example 2 (Separation of Concerns)

- a controller that sends a message for an action to be initiated and then waits for that action to have started (or completed) before continuing (a `ROS` example)
  - structure of the message
  - we will send the desired speed
  - and get the actual speed back

```
int32 desiredSpeed
---
int32 reportedSpeed
```

THE MESSAGE

Example 2 (Separation of Concerns)

- a controller that sends a message for an action to be initiated and then waits for that action to have started (or completed) before continuing (a `ROS` example)
  - copies the desired speed back, as an acknowledgement of the speed that it is aiming at
  - sensor and actuator software modules should be kept separate

```
bool set(speed_controller::Speed::Request  &req,
         speed_controller::Speed::Response &res)
{
  res.reportedSpeed = req.desiredSpeed;
    // <set actuator to desiredSpeed> //
  return true;
}
```

THE SERVER

Example 2 (Separation of Concerns)

- a controller that sends a message for an action to be initiated and then waits for that action to have started (or completed) before continuing (a `ROS` example)
  - assume that the actuator module can easily access sensor data
  - reported speed would represent the speed when the command was issued, not when the actual speed was reached

```
bool set(speed_controller::Speed::Request  &req,
         speed_controller::Speed::Response &res)
{
  res.reportedSpeed = req.desiredSpeed;
    // <set actuator to desiredSpeed> //
  return true;
}
```

THE SERVER

Example 2 (Separation of Concerns)

- a controller that sends a message for an action to be initiated and then waits for that action to have started (or completed) before continuing (a `ROS` example)
  - add further complexity to the actuator module by making it wait until the desired speed is reached
  - How long should the actuator wait for the target speed to be reached?
  - Do we queue or let other call enter?

```
bool set(speed_controller::Speed::Request  &req,
         speed_controller::Speed::Response &res)
{
  res.reportedSpeed = req.desiredSpeed;
    // <set actuator to desiredSpeed> //
  return true;
}
```

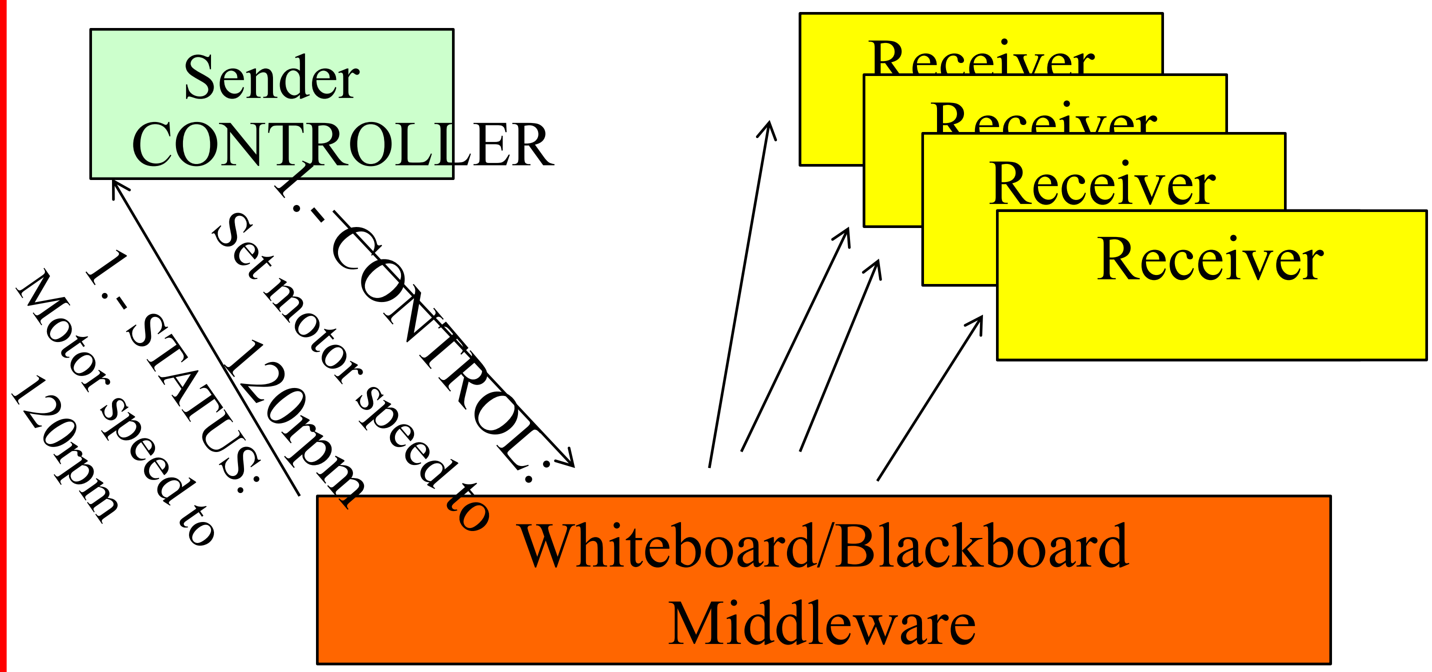
THE SERVER

*The **PULL** model is better*

- asynchronous messages are used,
  - a `Control` message for the actuator and
  - a `Status` message for the sensor
- Both the actuator and the controller components us the `PULL` paradigm to query their corresponding message slots.
- Decoupling the sensors modules from the actuator modules

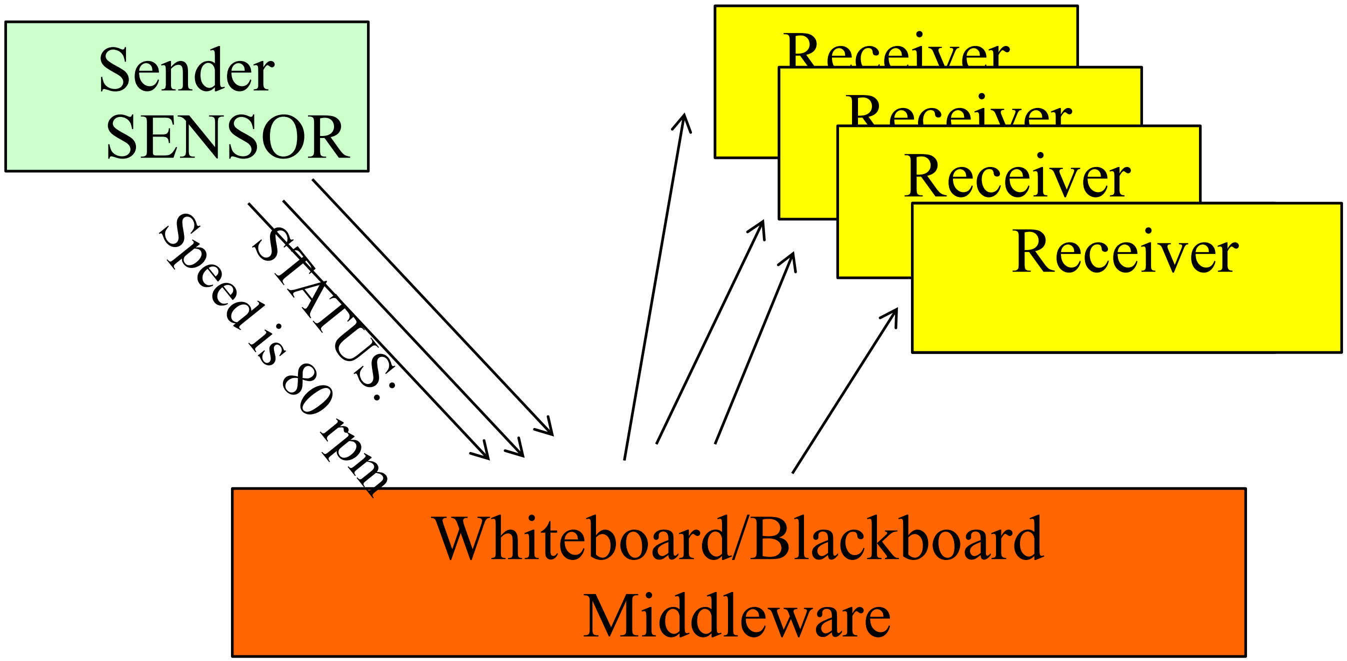
Blackboard architecture (CONTROL/STATUS)

- Controller on an actuator
  - Control message followed by monitoring status message



Blackboard architecture (STATUS)

- Information producer, regularly post STATUS



Locomotion

- The other big avenue of robotics
  - as opposed to manipulation

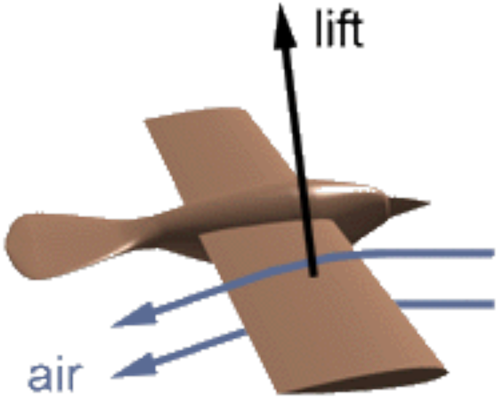
- Modern definition of Robot
  - An autonomous system which exists in the physical world, can sense its environment and can act on it to achieve goals.

Locomotion – as found in nature

Type of Motion	Resistance to motion	Basic kinematics of motion
Flow in a channel	Hydrodynamic forces	Eddies
Crawl	Friction forces	Longitudinal vibration
Sliding	Friction Forces	Transverse vibration
Running	Loss of kinetic energy	Oscillatory movement of a multilink pendulum
Jumping	Loss of kinetic energy	
Walking	Gravitational forces	Rolling of a polygon

Flows in nature and mankind

- gliding



- sailing

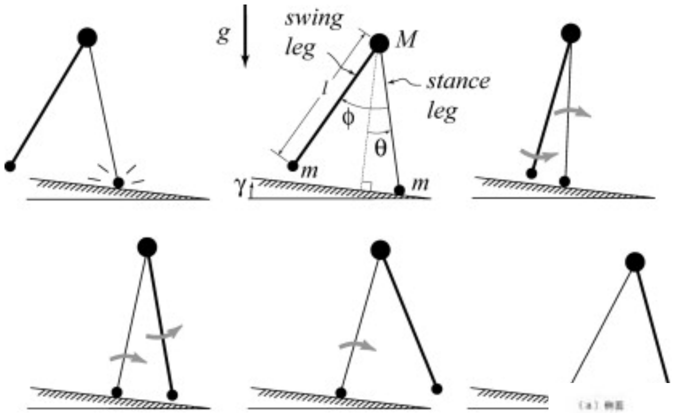
Nature

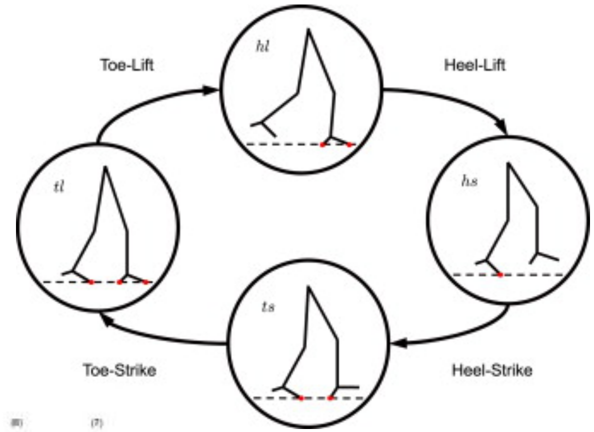
- Locomotion adapted to environmental characteristics
  - context, size
  - but does not employ wheels (bipedal walking)
- Difficult to imitate technically
  - technical systems today use wheel on the ground
    - stable (the robot, with implications to sensors)
  - propellers on fluids
  - legged motion is still mostly a research topic.

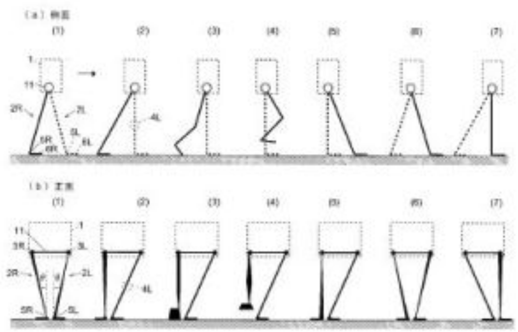
Robots and people

Biped walking mechanism

- Vast diversity of approaches
  - not far from rolling, as step is smaller
  - rotating joint not invented by nature
  - needs dynamic gait, and work against gravity







Walking or rolling?

- Trade-off
  - speed
  - number of actuators
  - structural complexity
  - control expense
  - energy efficient
    - terrain (flat ground, soft grounds, climbing …)
  - movement of the involved masses
    - walking / running includes up and down movement of the Center Of Gravity (COG)
    - some extra losses

Why do we walk? When do we go on wheels?

- Walk is very diverse, flexibility of the environment and the terrain
  - good complex terrain,
  - speeds less than 10km/hrs
- Travel on wheels is fast
  - but must be flat, regular
  - efficient for speeds over 100km/hrs

Interaction with the environment

- interaction forces and the mechanisms (effectors) and actuators that generate them
- STABILITY
  - number of contact points, center of gravity, static/dynamic stabilization, terrain inclination
- CHARACTERISTICS OF CONTACT
  - contact point vs contact area, angle of contact, friction
- TYPE OF ENVIRONMENT
  - structure, medium (water, air, soft ground vs hard ground

Robots with legs

- The fewer the legs the bigger the stability challenge
  - stability with point of contact
    - at least three legs are required for static stability
  - stability with surface contact
    - at least one leg is required
- During walking (some --- usually half) of the legs are lifted
  - reducing (loosing) stability?
- For static walking, at least 4 (or 6) legs are required

Soft implication

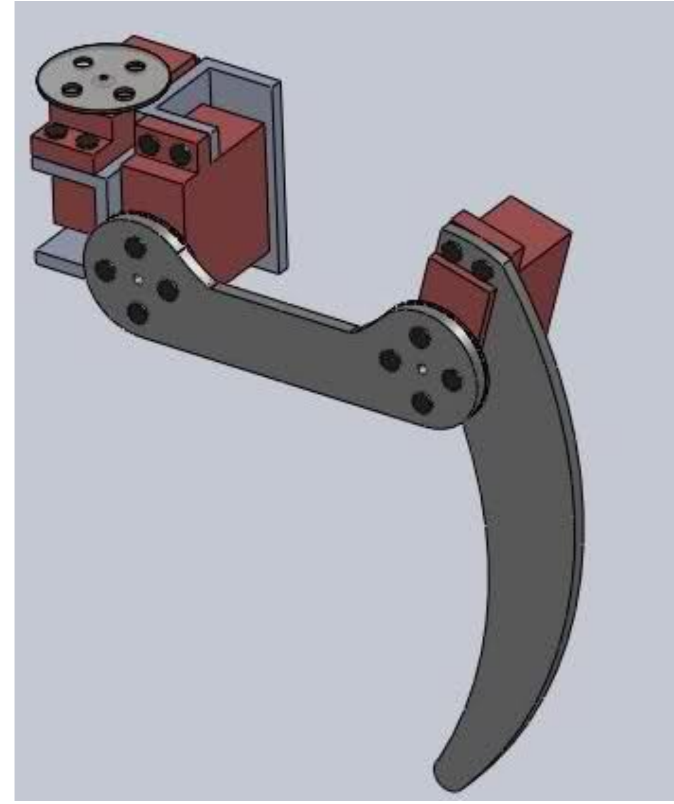
- Less legs require more sophisticated controls
  - bipeds – humans
  - four legs mammals
  - four legs reptiles
  - insects 6 or more legs
- What were Grey Walters tortoise?
- What are Braitenberg's Vehicles?

Number of joints of Each Leg --- Degrees of Freedom (DOF)

- A minimum of two DOF is required to move a leg forward
  - a lift and a swing motion
  - sliding-free motion in more than one direction not possible
- **Leonardo Da Vinci's Lion Robot for the King of France, Year-1515**
  - **http://www.youtube.com/watch?v=7jBkwCWxaic**

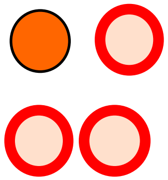
Legs

- Three (3) DOF for each leg in most cases
- 4th DOF for the ankle joint
  - might improve walking and stability
  - additional joint (DOF) increases complexity of the design and locomotion control



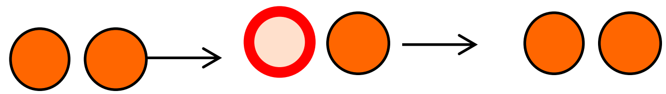
GAITS – number of distinct event sequences

- 4 state of two legs (biped) robot
  - both legs down
  - left up, right down
  - right up, left down
  - both up
- States of a four legged robot
  - 16 states
    - left front and 3 others up



- In general, 2^k states, for *k* legs

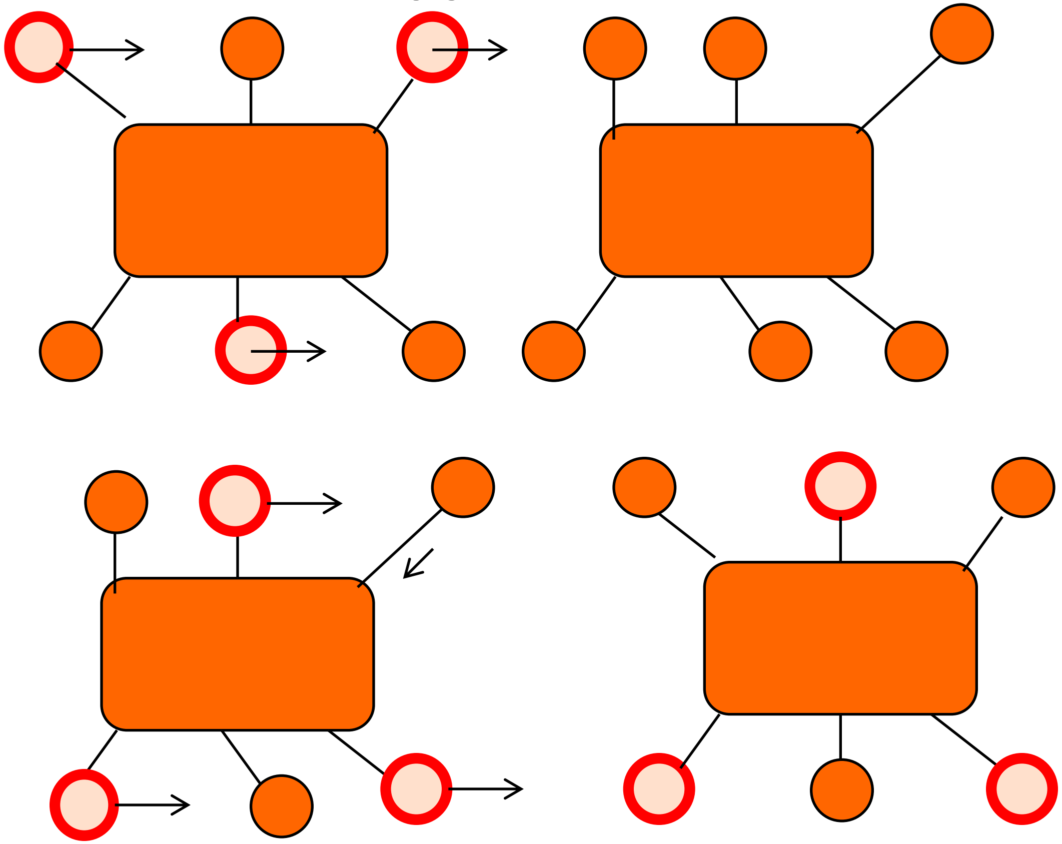
GAITS – number of distinct event sequences

- sequences of two legs (biped) robot
  - turning on a right leg 
- How many sequences with 2k objects where I don't repeat.
  - and I take away the symmetries of the robot
- At least $(2^k\text{-}1)!$
  - the more legs the more complexity of the possible gaits!
    - the less legs the more complexity of the motion

Robots and people

Most obvious gaits

- 6 legs, static, a triangle supports the COG at all times



Illustration

- **Six-Legged Walking Robot**
  - http://www.youtube.com/watch?v=GyjzBW-HQrg
- **4 Legged Walking/Crawler Robot 1**
  - **http://www.youtube.com/watch?v=GayaZQ2i-wc**
  - **Quadruped Robot: sQ1 Trot Gait(Forward)**
    - **http://www.youtube.com/watch?v=-ZgkTG-ZquM**

Robocup Aibo legged league

- MiPAL goals
  - **MiPal First Ever Goal**
    - **http://www.youtube.com/watch?v=egIOvgqoVEU&feature=player_embedded**
  - **MiPal Aibo Goals JPN**
    - **http://www.youtube.com/watch?v=lWMHzMWUkJ8&feature=player_embedded**

Dynamic Walk vs Static walk

Statically stable

- Bodyweight supported by at least 3 legs
  - COG in support area
- Even if the joins 'freeze' instantaneously, the robot will not fall
  - trade-off prefers safety over speed

Dynamic walking

- Robot will fall if not continuously moving
  - motor-bike analogy
- Less than three legs can be in ground contact
- fast, efficient, but demands for actuation and control

Dynamic walking with Four Legs

- Quadruped
  - **Boston Dynamics Big Dog (new video March 2008)**
    - **http://www.youtube.com/watch?v=W1czBcnX1Ww**

The progress in humanoid/ 2-legged walking

Stiff robots are dangerous

- **World's Most Dangerous Robot**
  - **http://www.youtube.com/watch?v=6Mq_BTzA1Hc**
- Investigate what are compliant robots
  - Most environments using robots do not allow human beings nearby

Passive Dynamic Walker

- **Passive Walking Robot Propelled By Its Own Weight** http://www.youtube.com/watch?v=rhu2xNIpgDE

Contrast of flexible / passive walk

- Efficiency
  - c_{mt} = | mech. energy|/(weight × distance traveled)
- Asimov
  - $c_{mt} = 1.6$
- Bipedal ETH
  - $c_{mt} = 0.31$
- Passive walker
  - $c_{mt} = 0.055$

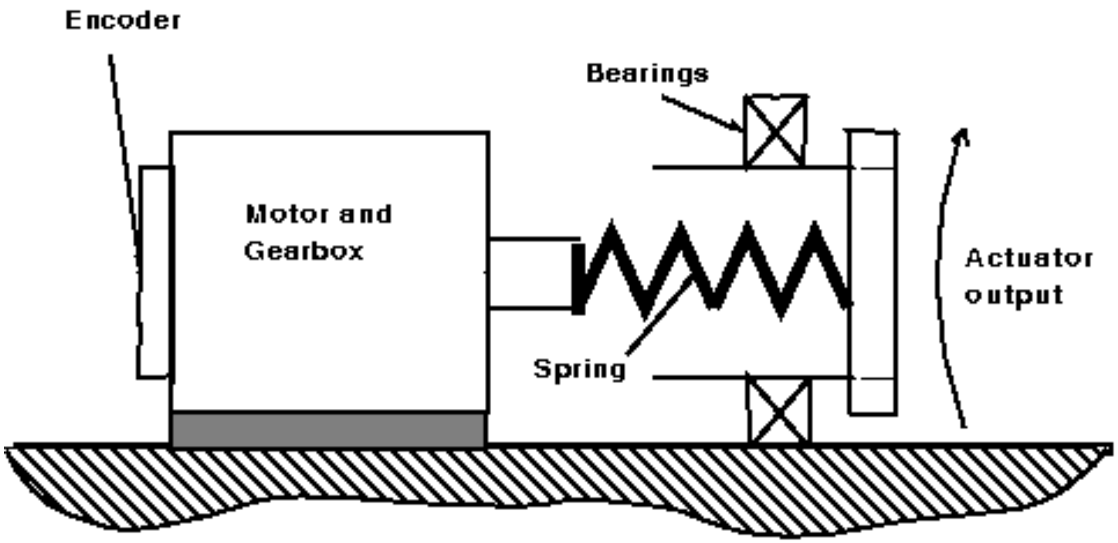
Robots and people

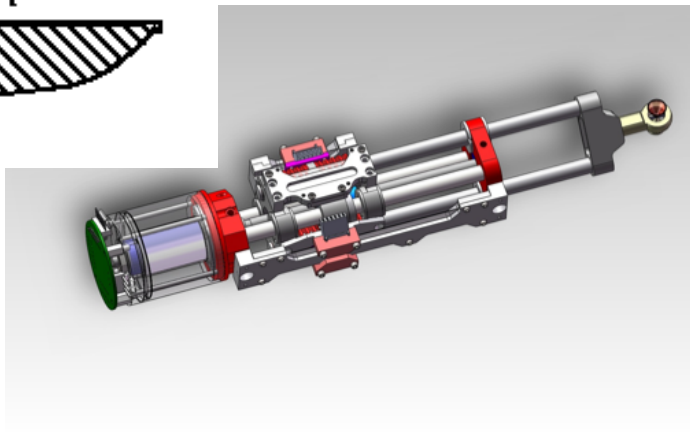
Research for dynamic walking

- Optimized gaits
  - use ‘elasticity’ as a storage for energy
  - allow locomotion for varying frequencies and speeds
    - different gaits gave different elements behave differently
- The energetically most efficient gait is a function of desired speed.

Series Elastic Actuator

- **The best five biped walkers**
  - http://mindtrans.narod.ru/walkers/walkers.htm





Efficient dynamic walking

- requirements
  - back-drivable to allow unimpeded natural dynamics
  - able to perform negative work
  - have a low inertia and gear ratio to keep the reflected inertia small
  - have an adjustable actuator compliance
  - highly efficient
- series elastic actuation
  - requires active energy consumption
  - not all the benefits of passive dynamics

Robots and people

Its is much easier on wheels

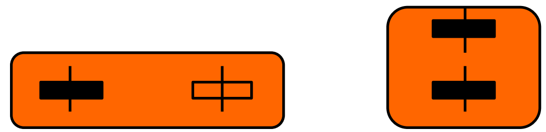
- Three wheels are sufficient to guarantee stability
  - More than 3 demands suspension
  - There are many types of wheels
    1. standard wheel
       - 2 DOF
    2. castor wheel
       - 3 DOF
    3. sweetish wheel
       - 3 DOF
    4. ball or spherical wheel
       - suspension issues

Characteristics of robots on wheels

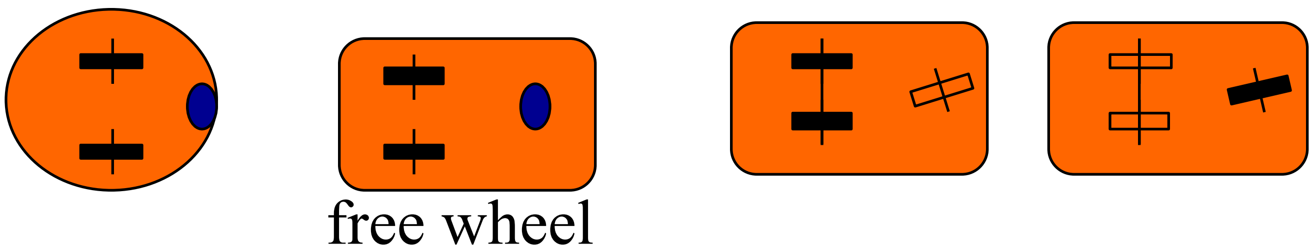
- Stability of the vehicle guaranteed with 3 wheels
  - if the center of gravity is within the triangle which is formed by the ground and the contact point of the wheels
  - stability improved with 4 or more wheels
    - however, the arrangements are hyper-static and require a flexible suspension system
  - bigger wheels allow overcome high obstacle
  - most arrangements are non-holonomic
  - Combining actuation and steering on one wheel make the design complex and adds additional errors for odometry

Different arrangements of wheels

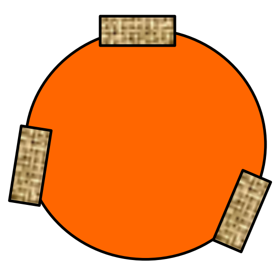
- Two wheels



  - COG below the wheels
- Three wheels



  - point of contact



Synchro Drive

- All wheels operate on synchronous actuator (one motor)
  - defines the speed
- All wheels are steered synchronously by a second motor
  - sets the heading of the vehicle
- **LEGO Mindstorms NXT, HOLO3CYCLE**
  - **http://www.youtube.com/watch?v=dqPMegizjwk**

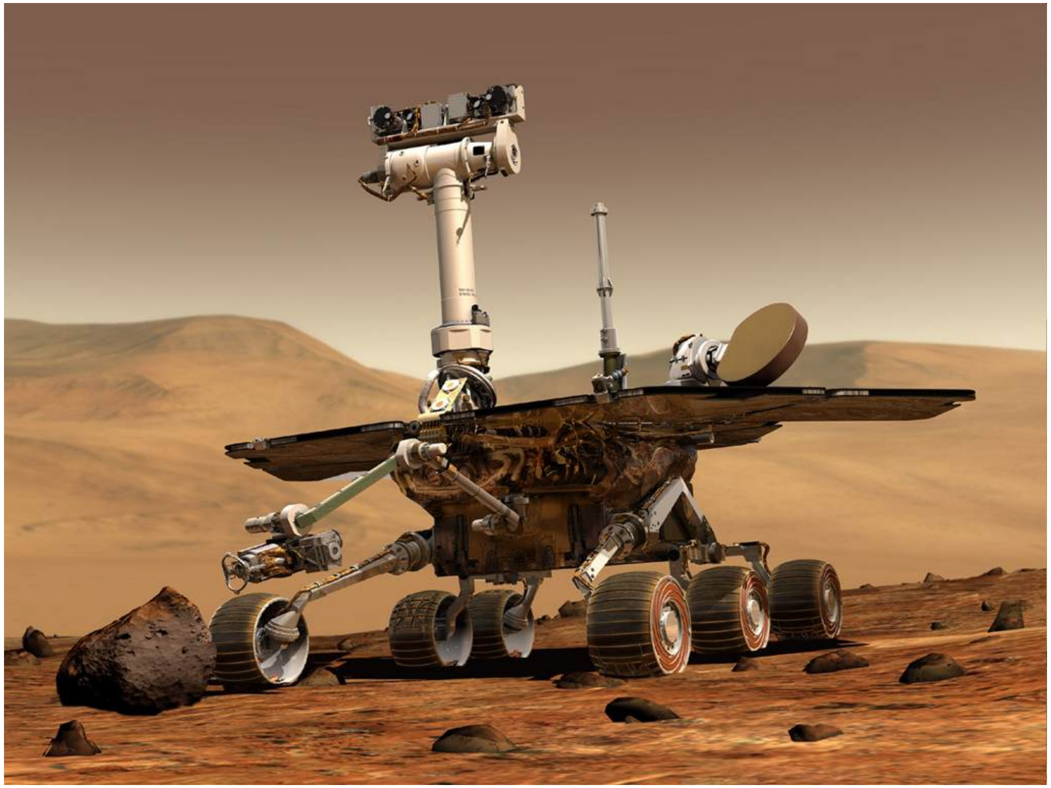
CMU Uranus. Omnidirectional 4 wheels

- Movement in the plane has 3 DOF
  - only 3 wheels can be independently controlled
  - perhaps better just 3 wheels

Robots and people

Other locomotion's

- Rovers
  - ie obstacles



other locomotion

- caterpillars
  - ie surface

Robots and people

other locomotion

- flying

Summary

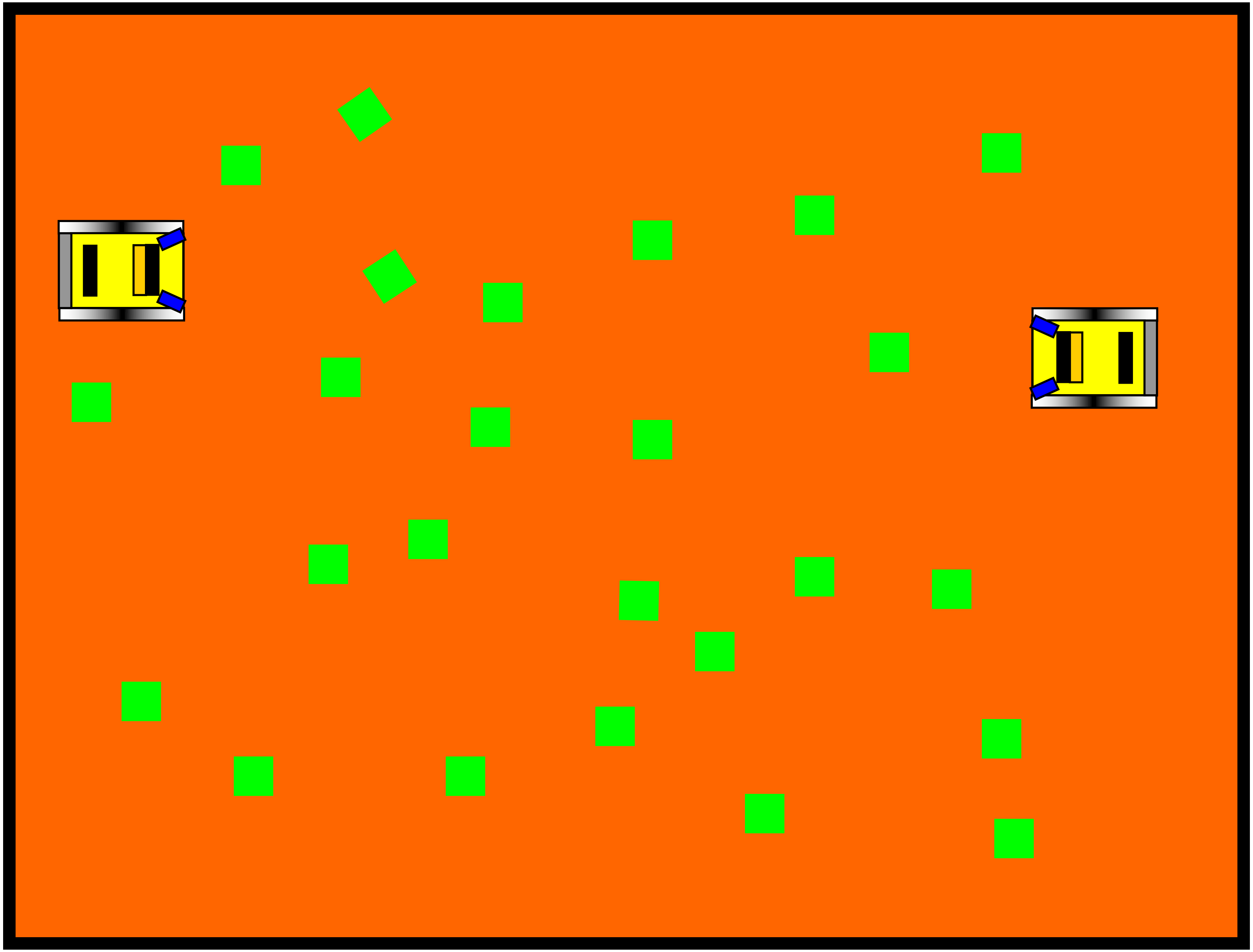
- Locomotion is challenging
- Legs are much more versatile for the environment but far more challenging
- Wheels are less versatile, but provide more stability.
  - May offer a challenge for non-holonomic robots

Practical Exercise

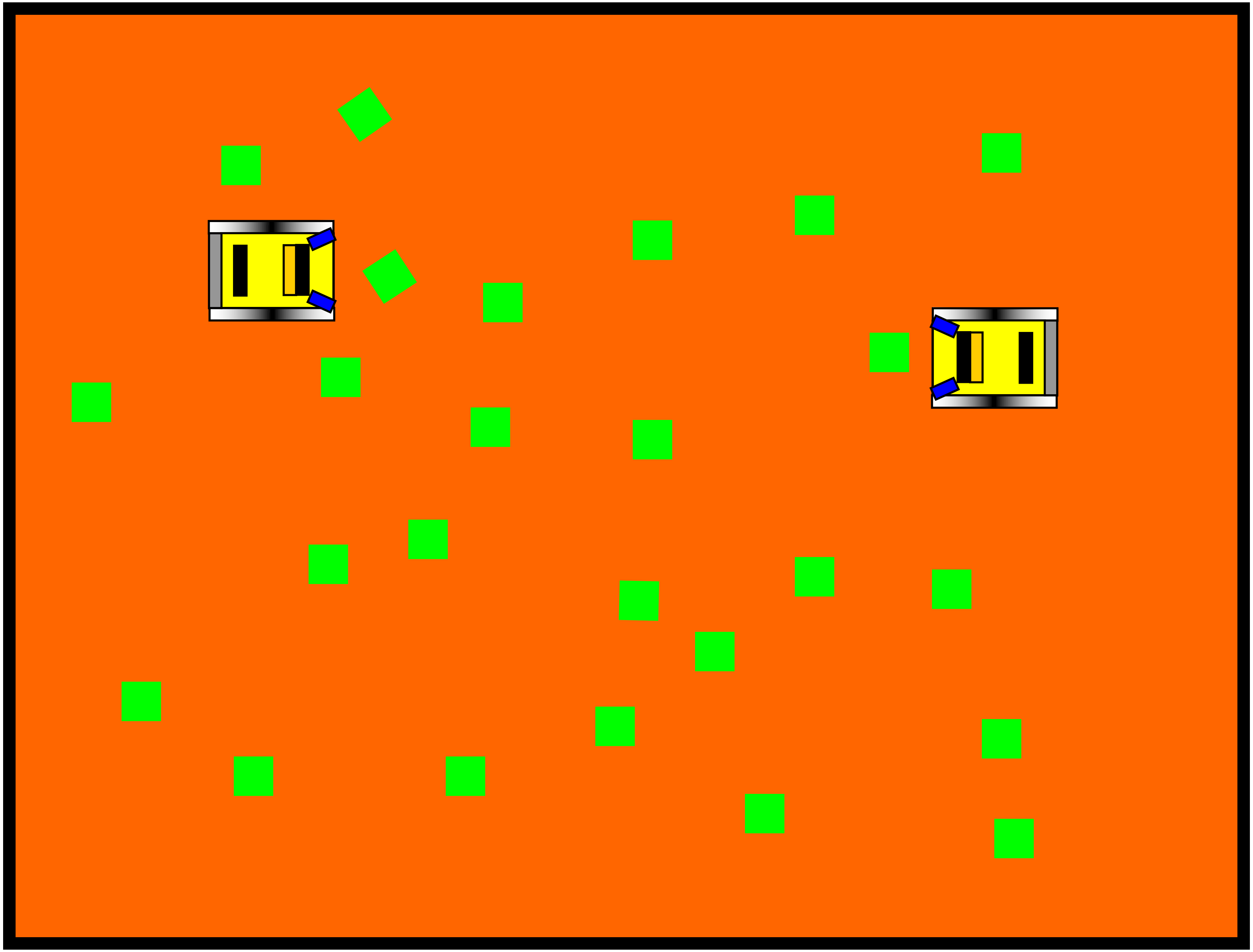
Emergent behavior

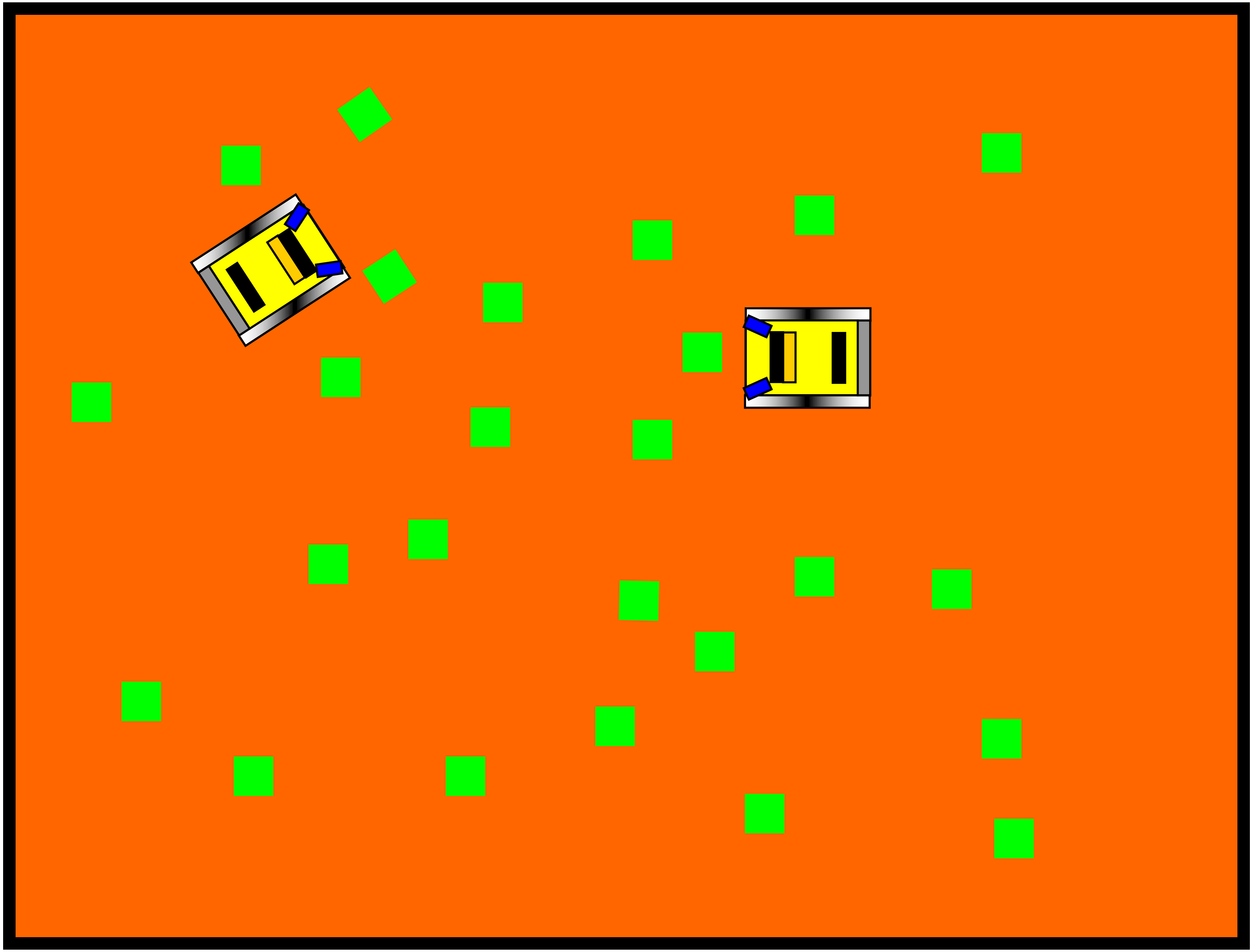
Myth 3:Look at the robot and you can discover what it is doing

Robots and people

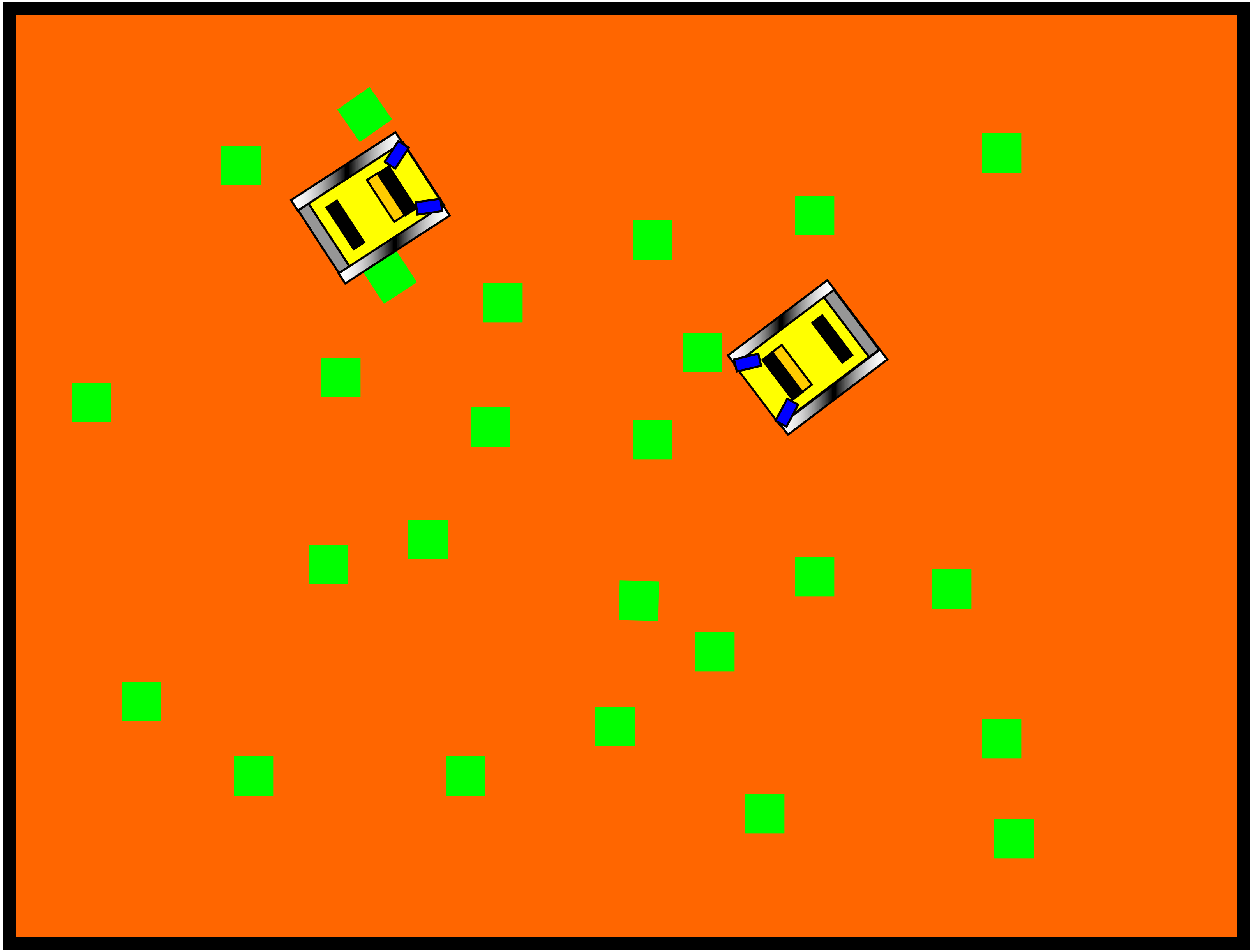


Robots and people

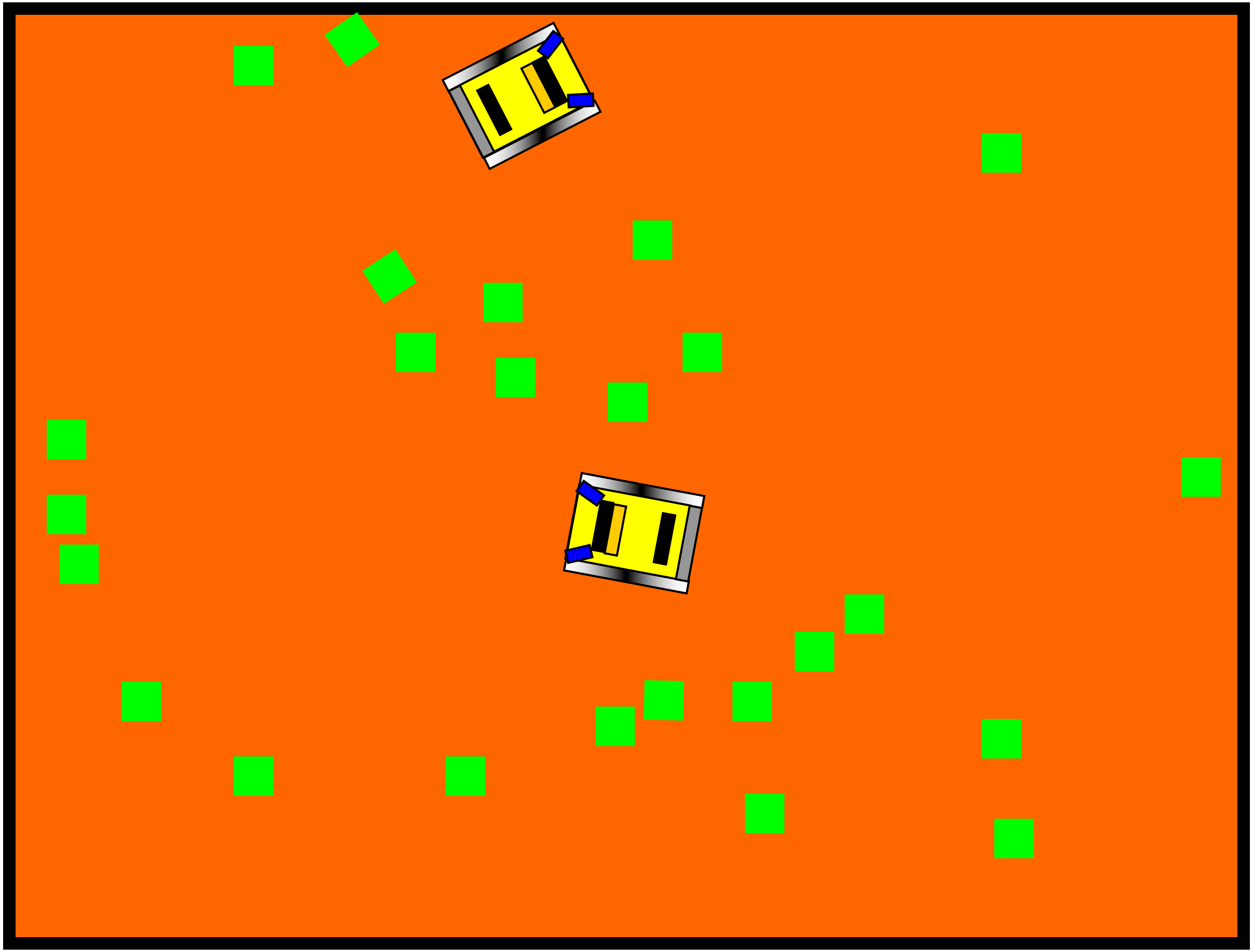




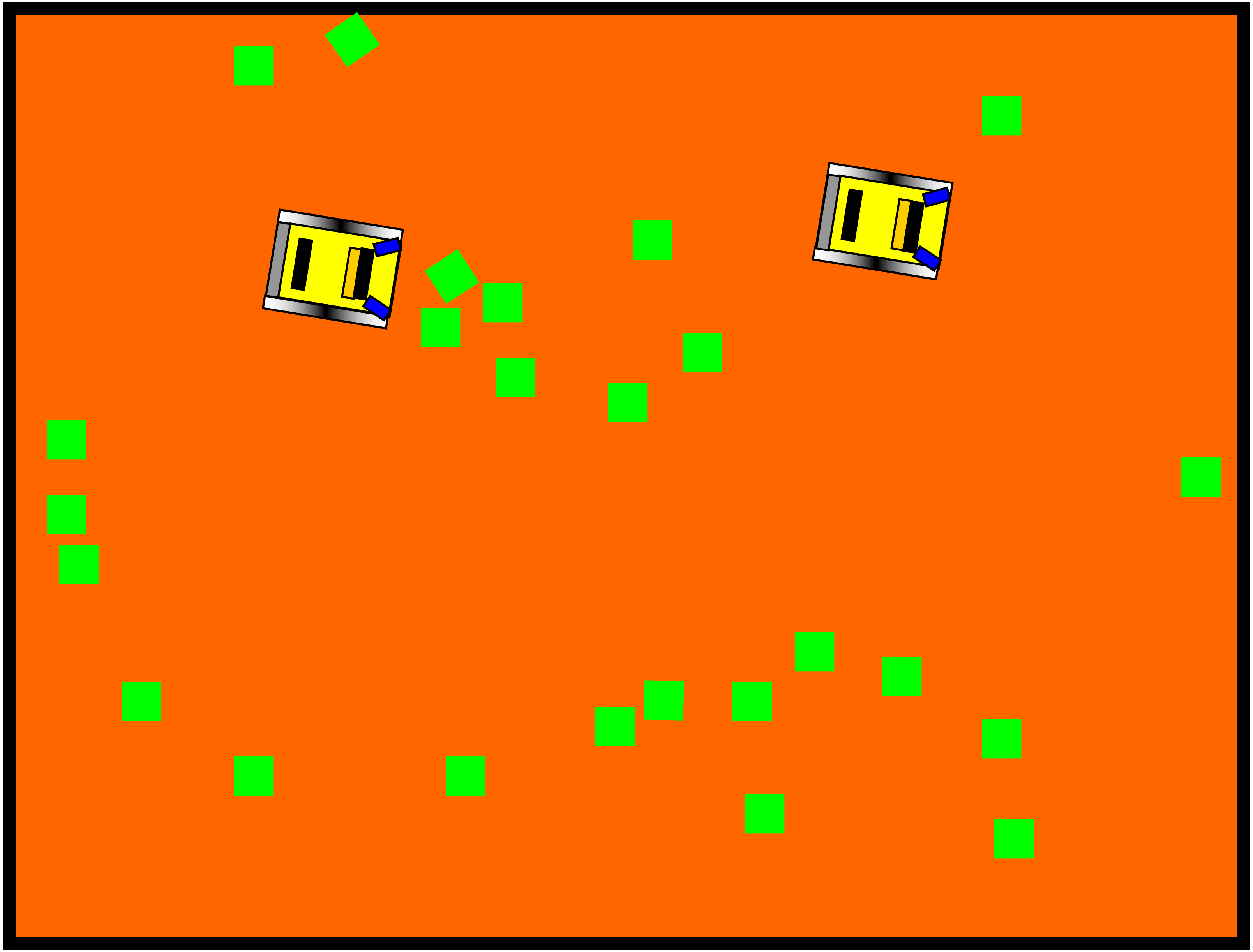
Robots and people



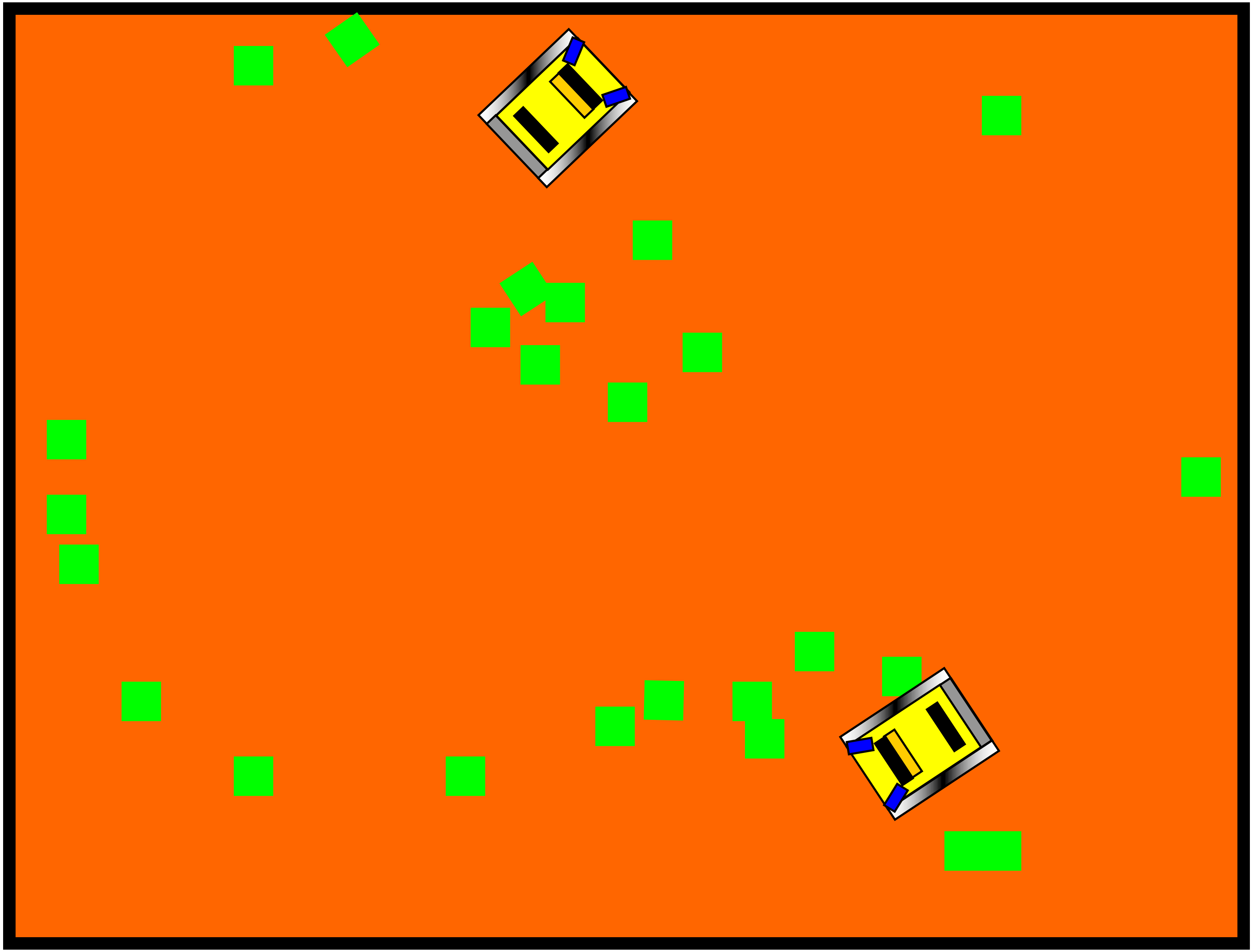
Robots and people



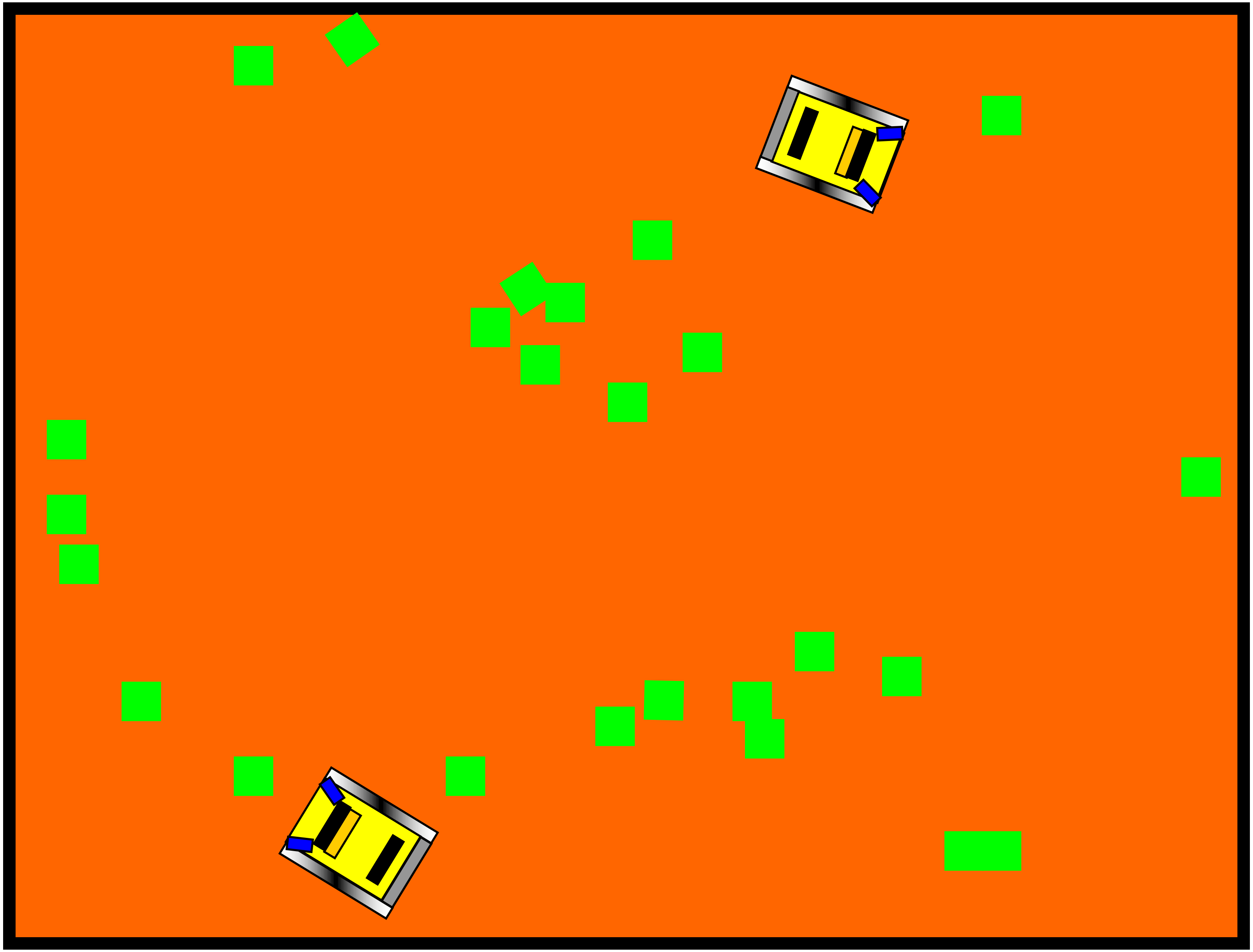
Robots and people



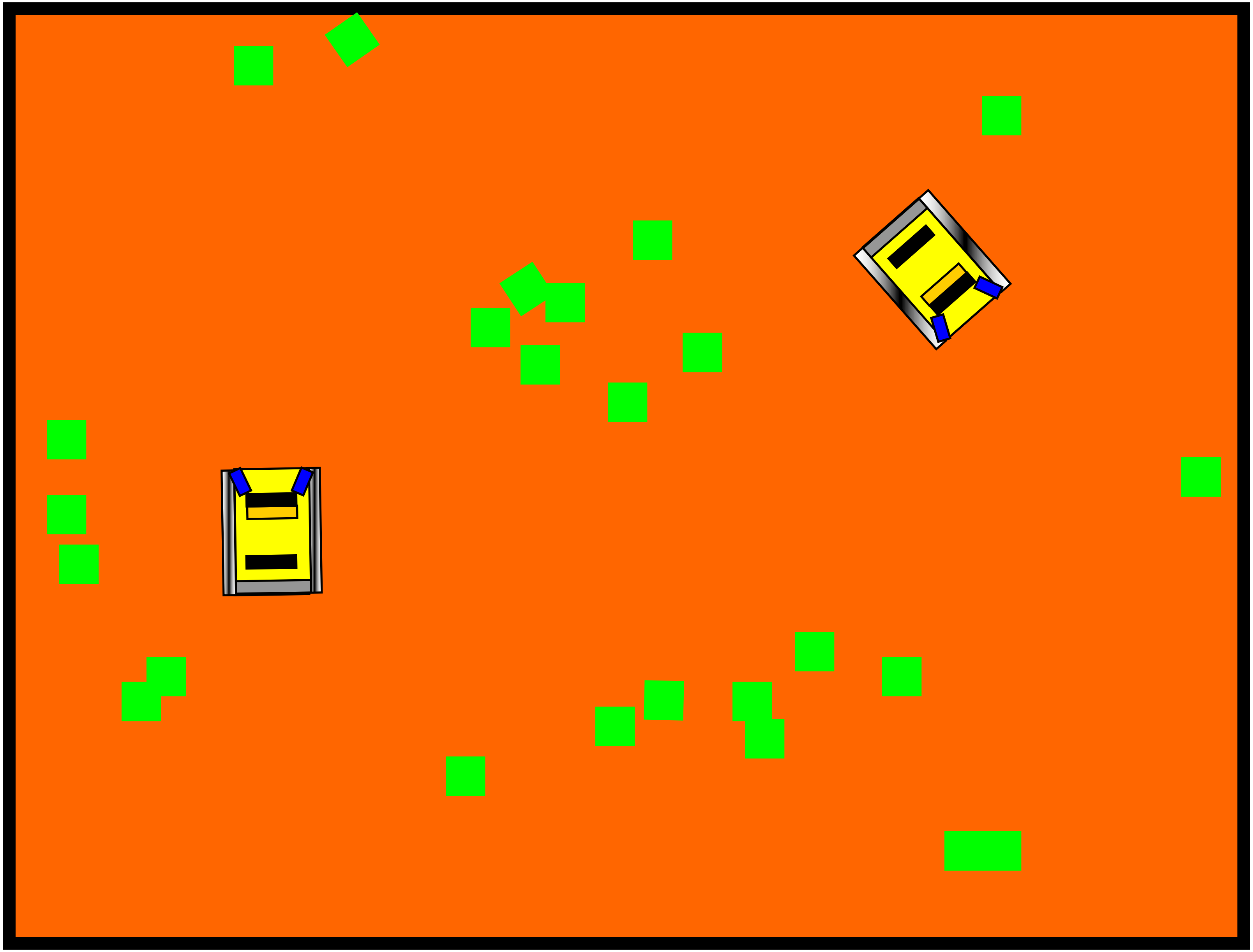
Robots and people



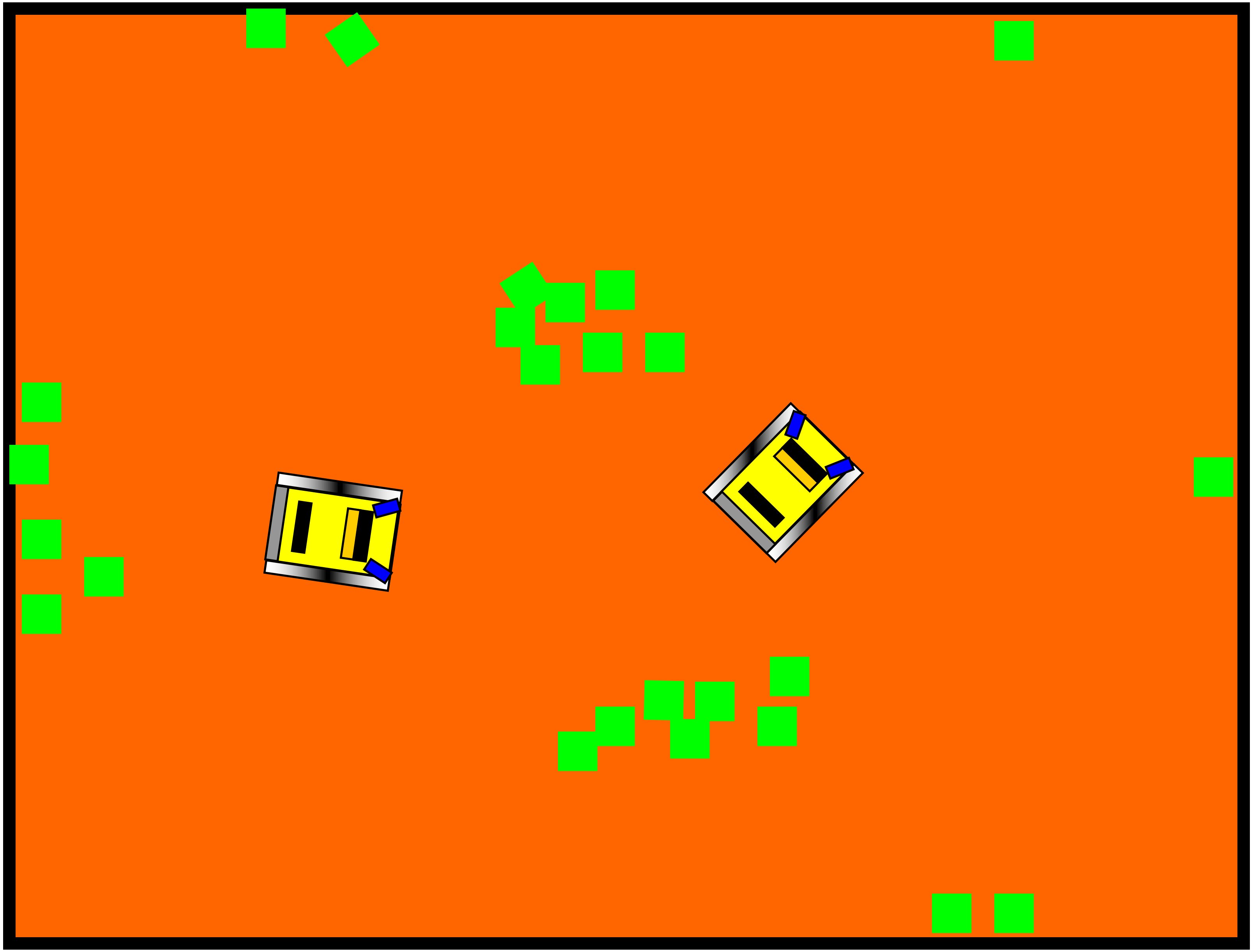
Robots and people



Robots and people

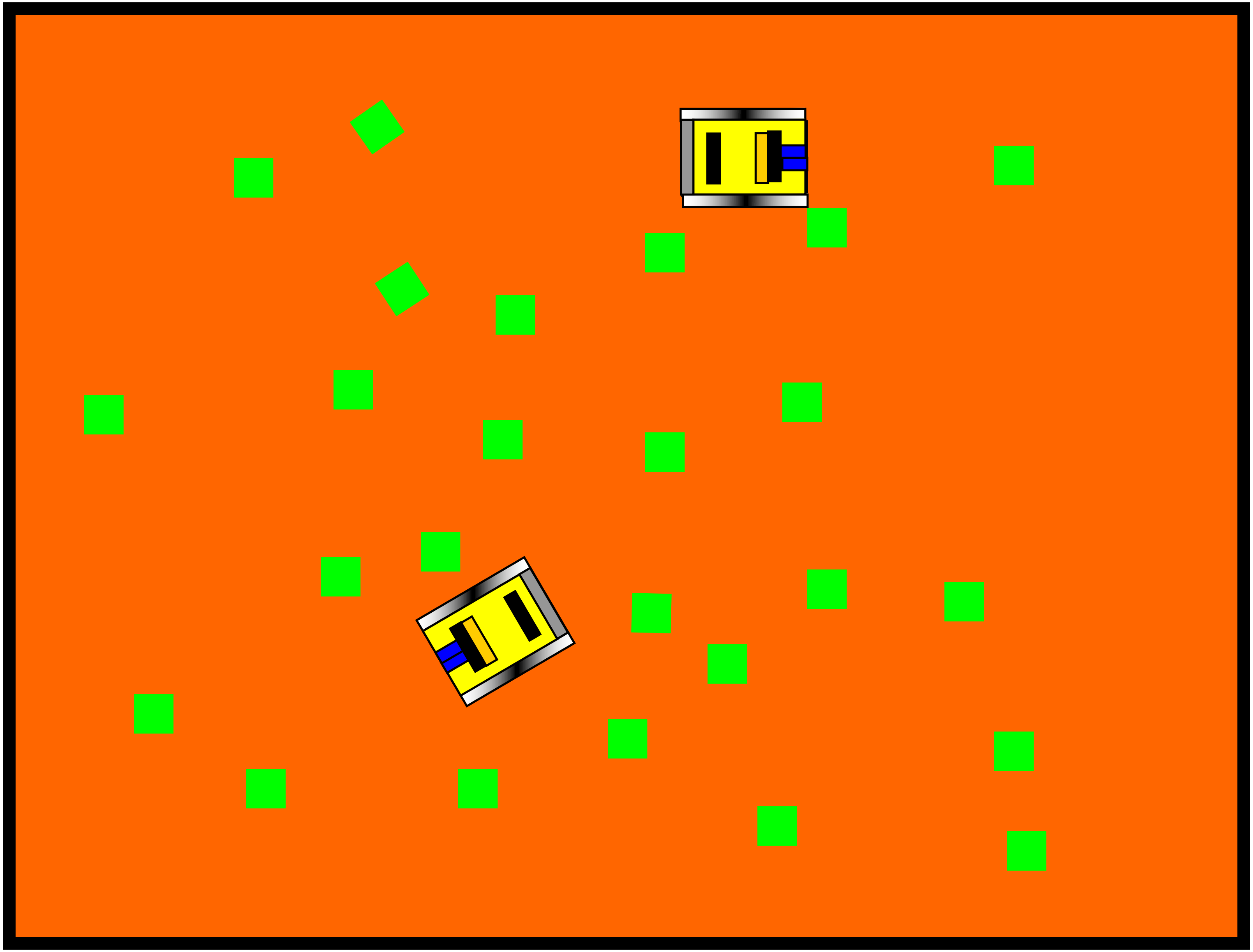


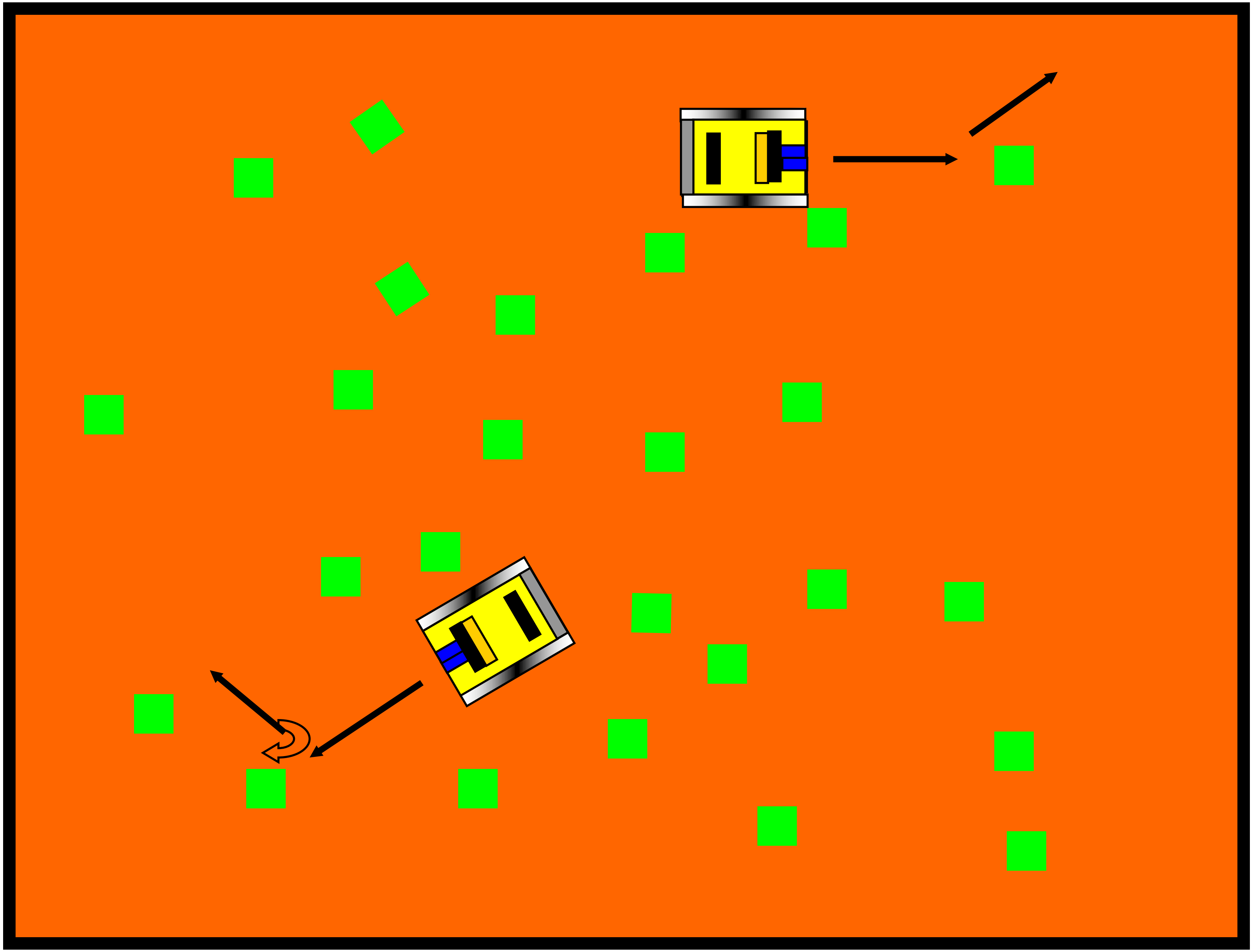
Robots and people



What is the team of robots doing?

Robots and people

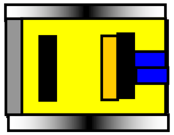




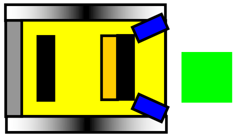
The behaviour (the program is the same)

1. Walk straight
2. If obstacle is visible, rotate so obstacle becomes invisible
3. If wall becomes visible, rotate away from the wall
4. Repeat

The hardware is different

- Sensor exactly in front 

- Sensors in an angle that allows an obstacle to be pushed without the robot noticing



  - Obstacles are moved until another one is close

Robots and people

THANK YOU