Performance
Is the room clean?

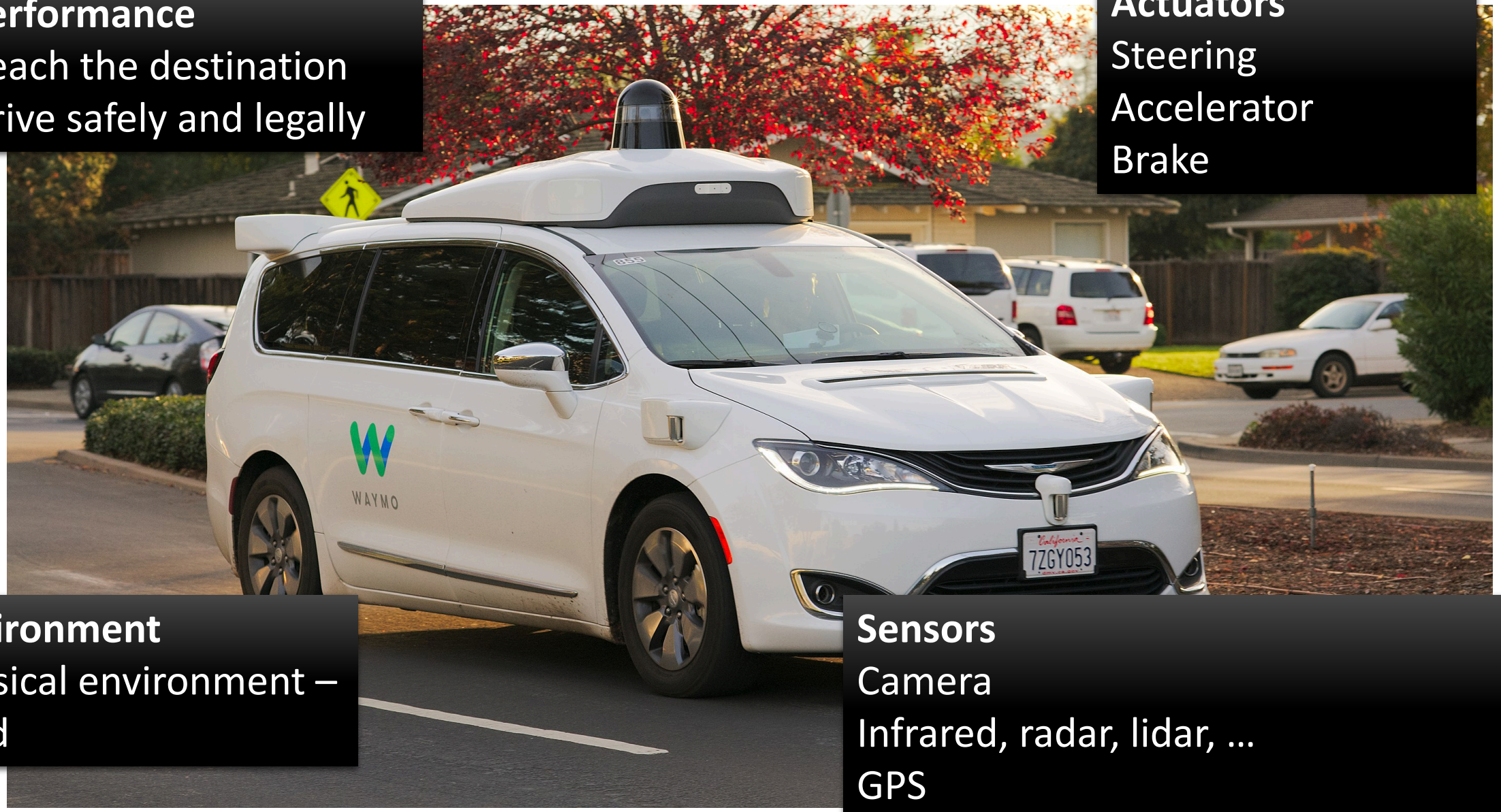
Actuators
Drive motors
Vacuum unit

Environment
Physical environment – a room

Sensors
Dirt sensor
Infrared sensors for walls/obstacles
Battery level
Dirt bin level

Performance
Reach the destination
Drive safely and legally

Actuators
Steering
Accelerator
Brake

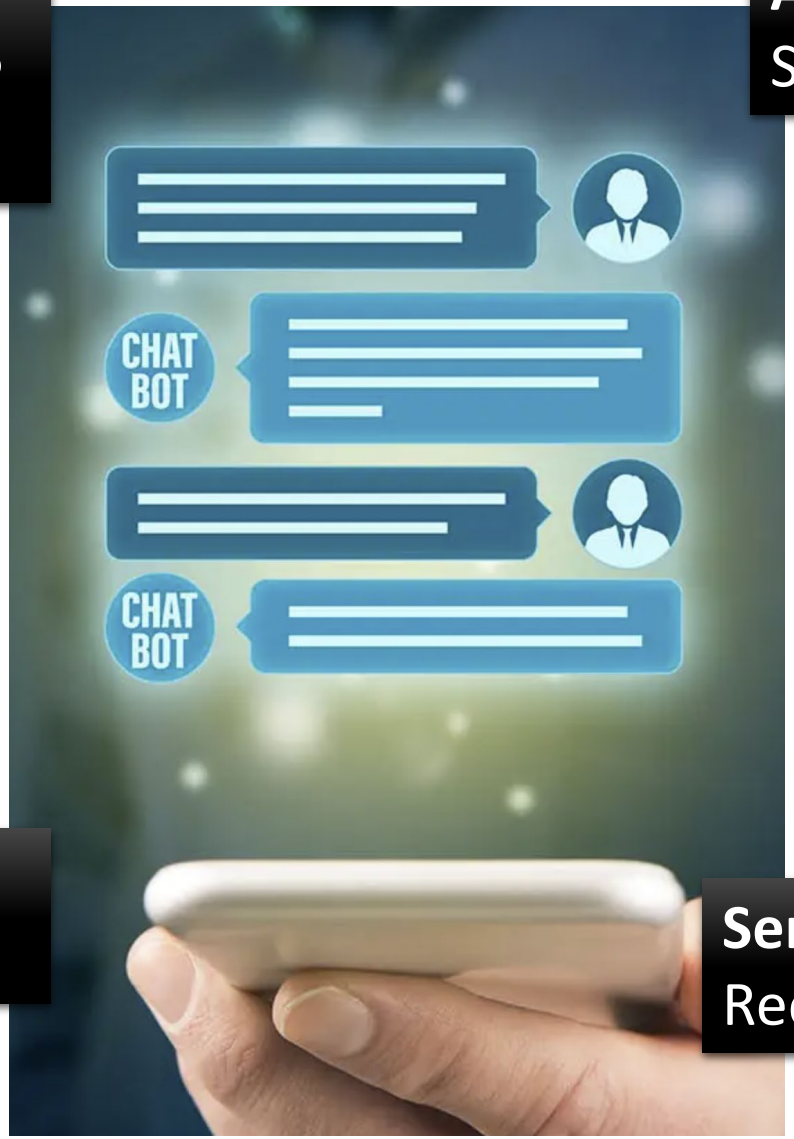


Environment
Physical environment – road

Sensors
Camera
Infrared, radar, lidar, …
GPS

Performance
Perform customer service?
Pass the Turing test?

Actuators
Send text message



Environment
Instant messaging app

Sensors
Receive text message

Performance
Win the game

Actuators
Make a move

Environment
Chess board

Sensors
Board state

Performance
Kill the player?
Provide convincing behaviour?

Actuators
CharacterController.Move()
Shooting
Playing animations

Environment
Game world

Sensors
Raycasts
Player / NPC positions
Level geometry

Performance
????

Actuators
Muscles
Vocal cords

Sensors
Eyes
Ears
Nose
Tongue
Skin

Environment
Real world

Performance
Win the game?
Have fun?

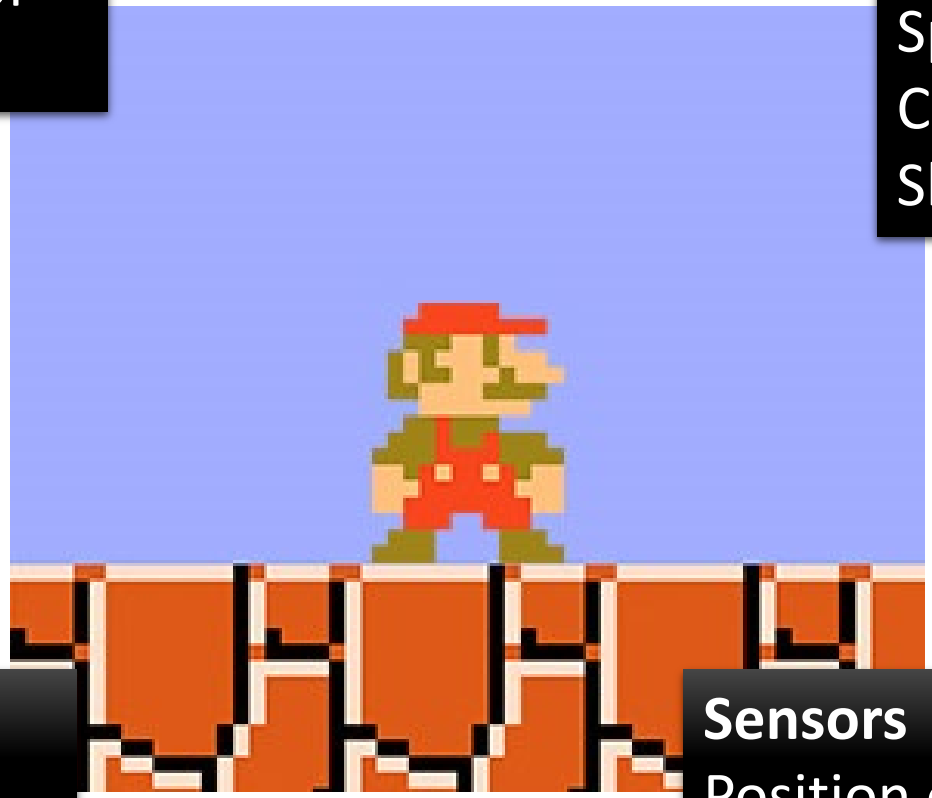
Actuators
Controller inputs

Environment
Game world

Sensors
Graphics
Sound
Haptic feedback

Performance
Reach the end of the level

Actuators
Move left/right
Jump
Sprint
Crouch
Shoot fireball

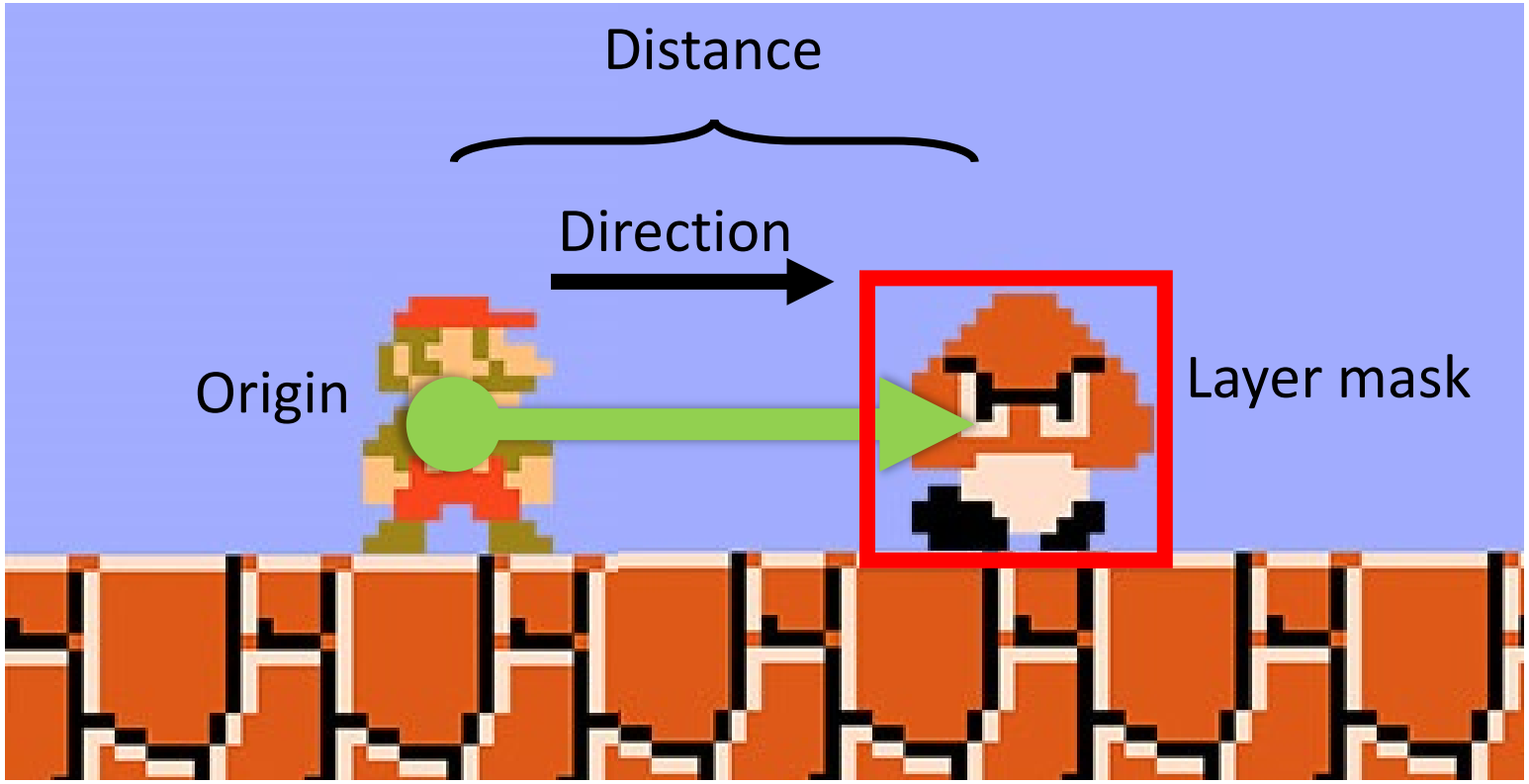


Environment
Game world

Sensors
Position of ground, enemies, coins, …

Sensing the world

- A common way for game agents to sense the world is using **raycasts**
- Cast a **ray (straight line)** from the agent out into the world
- Test if the ray hits a **collider**
- Unity: `Physics.Raycast` (for 3D), `Physics2D.Raycast` (for 2D)


Distance
Direction
Origin
Layer mask

Implementing AI to play Mario

- Fork and clone the **sample project** (link on LearningSpace and in chat)
- Load the **Main Menu** scene
- Formulate an AI controller as a set of if-then rules
- Code your AI logic into the `Update` method on the **AIMarioController.cs** script
- Let's see whose Mario can get the furthest!