# Creating a Dogfight Agent


Mediamatics / Data and Knowledge Systems group

Faculty of Electrical Engineering, Mathematics and Computer Science

Delft University of Technology, The Netherlands

Creating a fully autonomous **agent** that can play in a dogfight (1-on-1 battle) in Microsoft Combat Flight Simulator (MSCFS)

# Contents

1. Introduction
2. Design
3. Results
4. Conclusions

# Introduction

# Background

In 100 years:

- More airplanes
- More capabilities
- High information load for pilots

➢ Intelligent Cockpit Environment (ICE) project

# ICE project

- Situation recognition
- Mission or flight plan monitoring
- Attack management
- Pilot workload monitoring

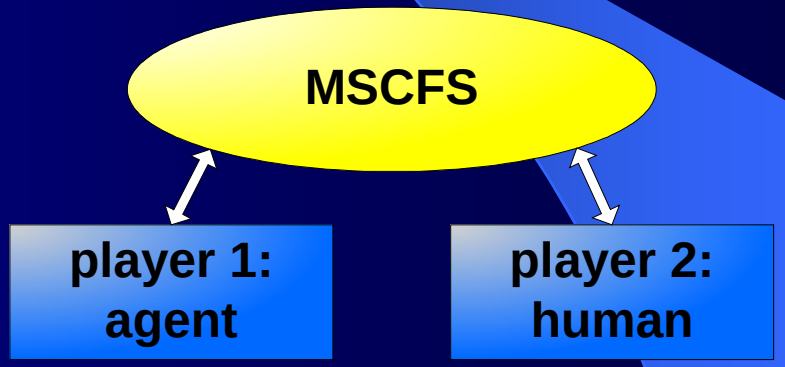
➢ Combine in one fully autonomous agent

# Project goal

Create an agent that can play a dogfight in MSCFS

Research:

- Literature study
- Model & design
- Prototype



# Requirements

- Fully autonomous
- No cheating
- Real-time working
- Possibilities for change and extensions

# Literature study

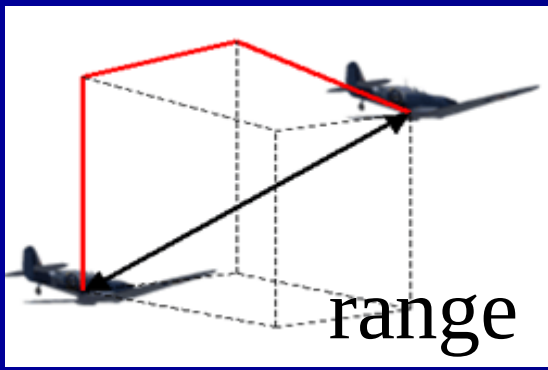
Areas:

- ICE project
- AI in aviation
- AI in computer games

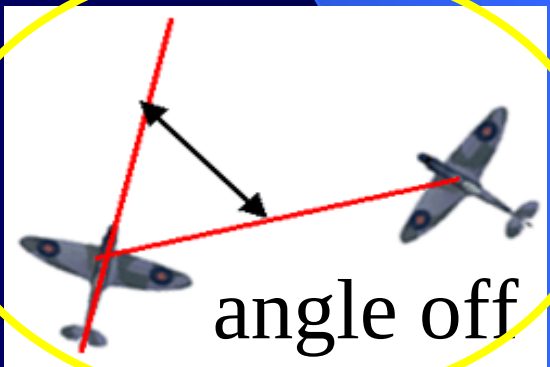
# Flight parameters

pitch

bank

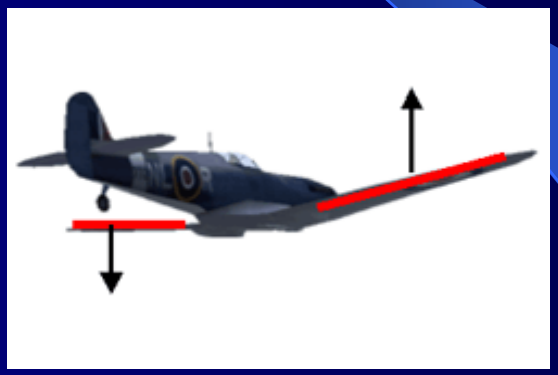






# Control parameters

elevator



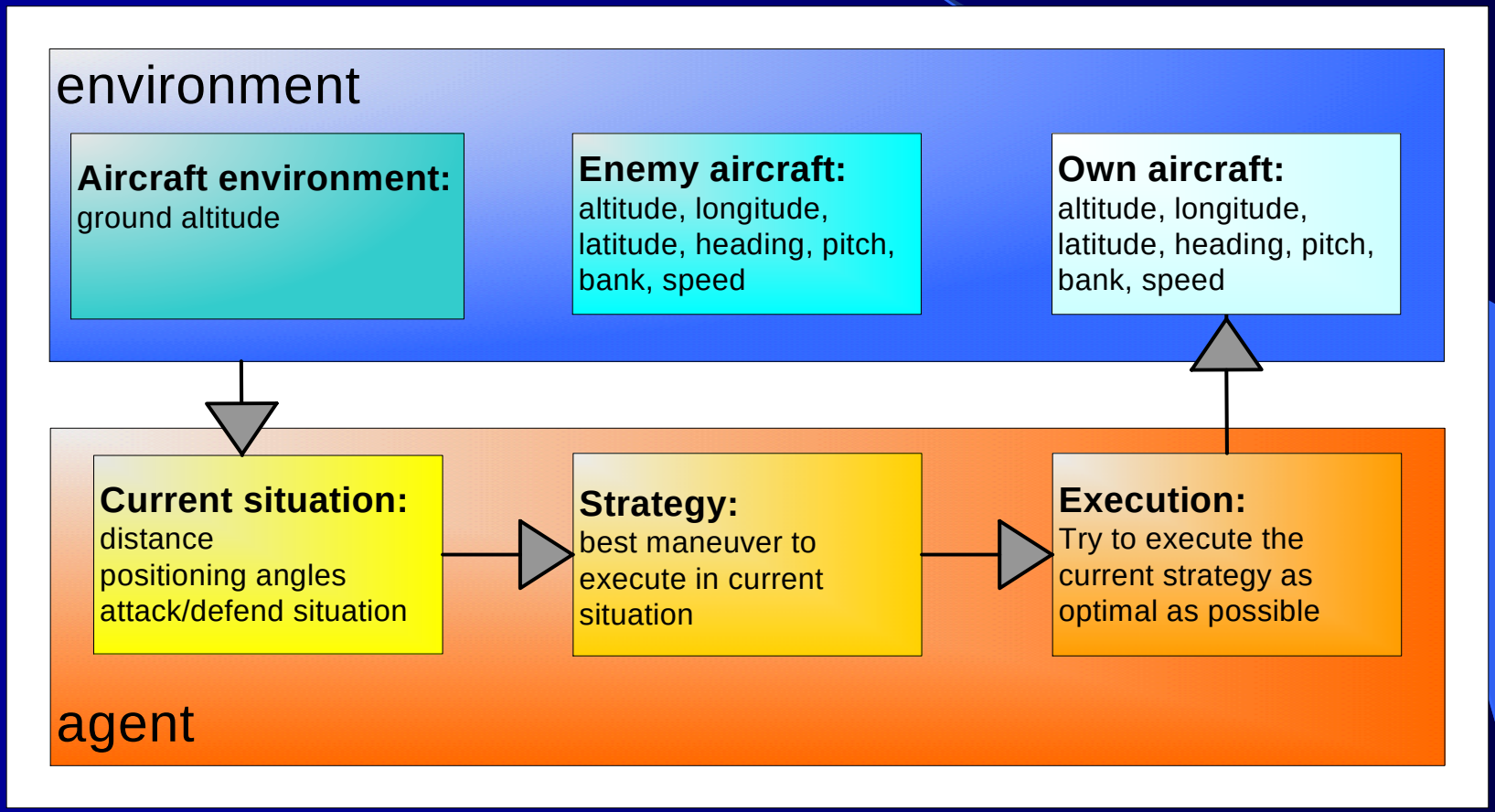
aileron

# Maneuvers

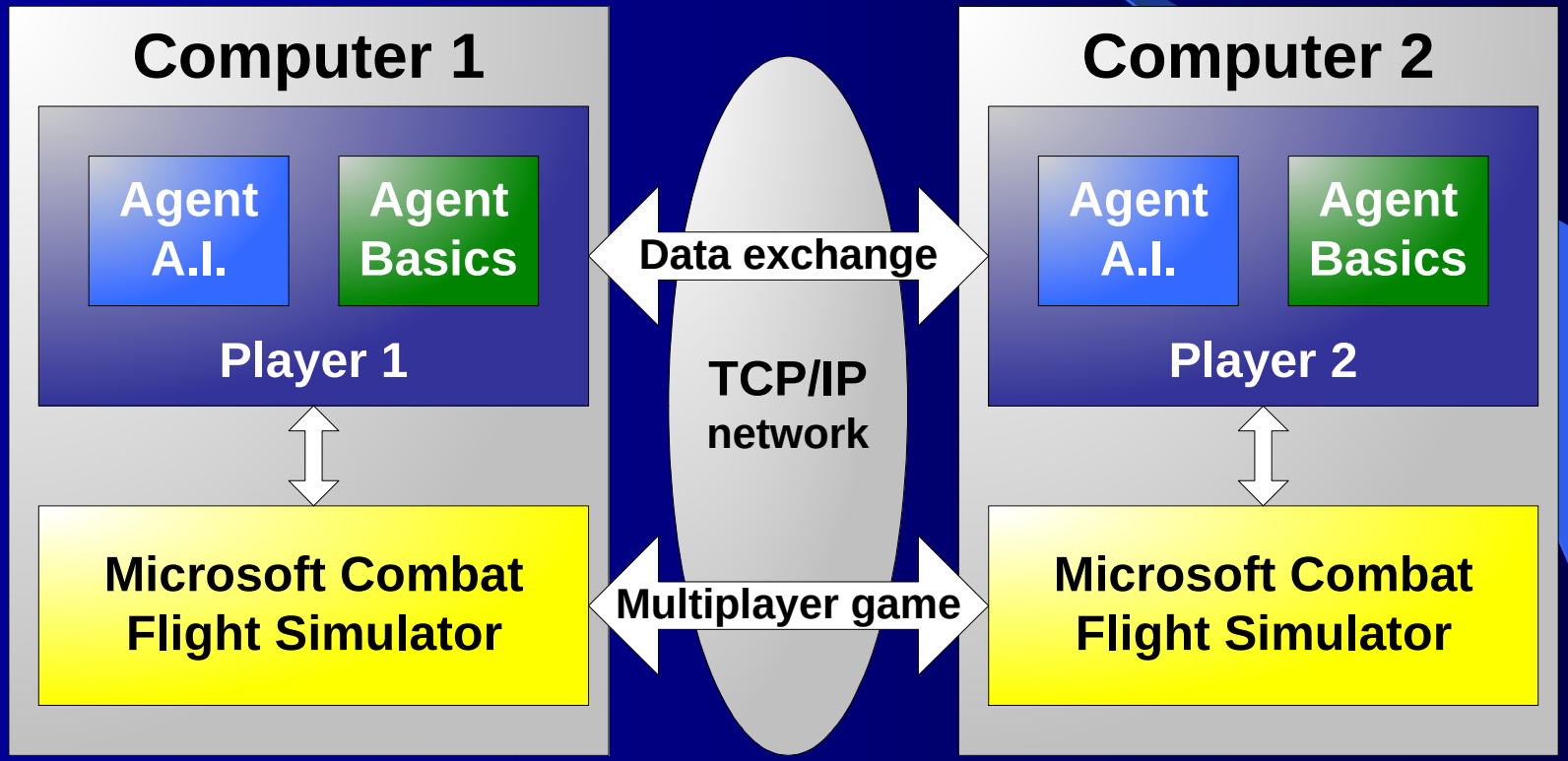
- Straight flight
- Turn (left, right)
- Extreme turn (left, right)
- Looping
- Split-S turn
- Immelmann turn

# Design

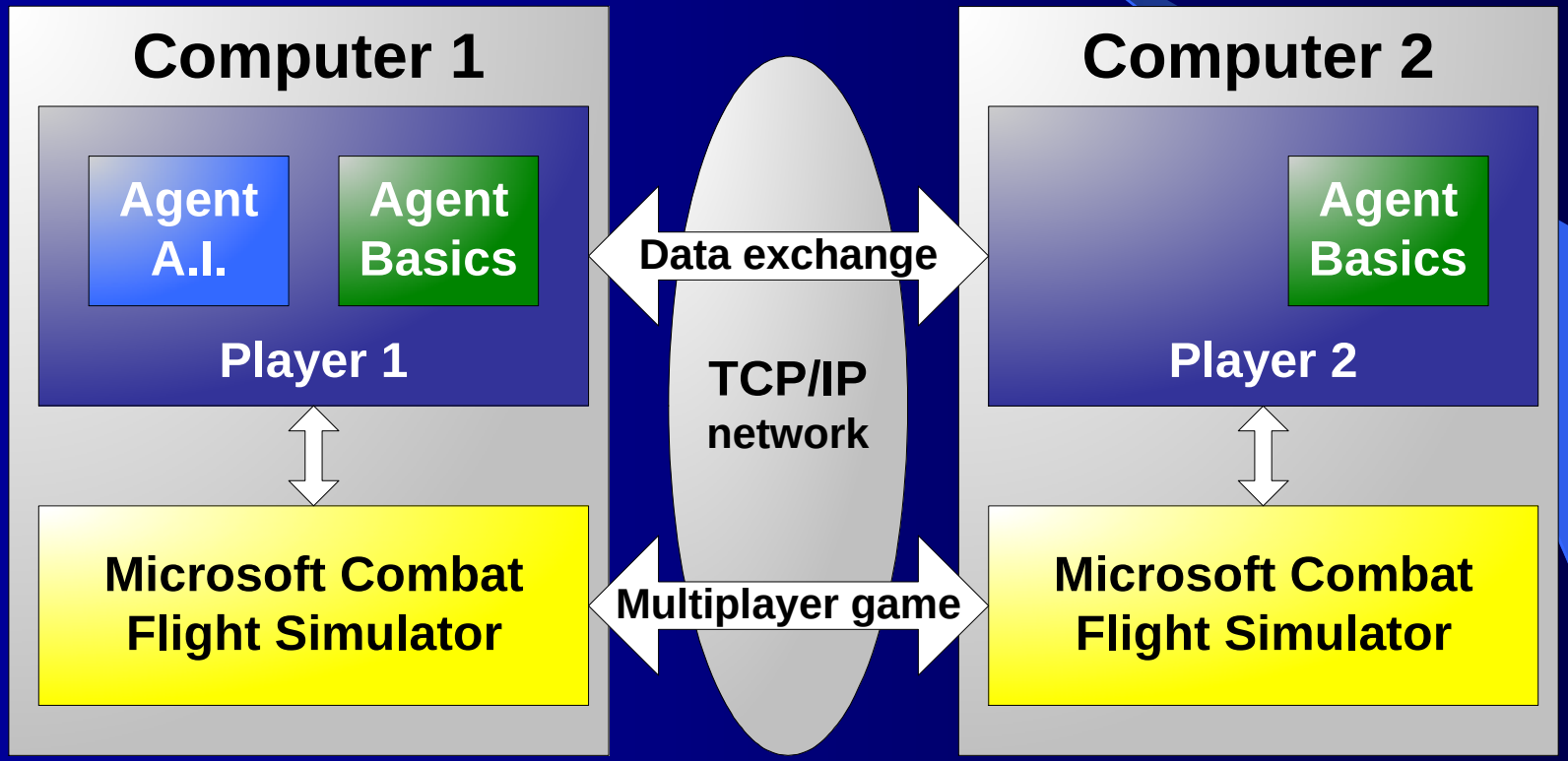
# Model


environment
Aircraft environment:
ground altitude
Enemy aircraft:
altitude, longitude, latitude, heading, pitch, bank, speed
Own aircraft:
altitude, longitude, latitude, heading, pitch, bank, speed
Current situation:
distance
positioning angles
attack/defend situation
Strategy:
best maneuver to execute in current situation
Execution:
Try to execute the current strategy as optimal as possible
agent

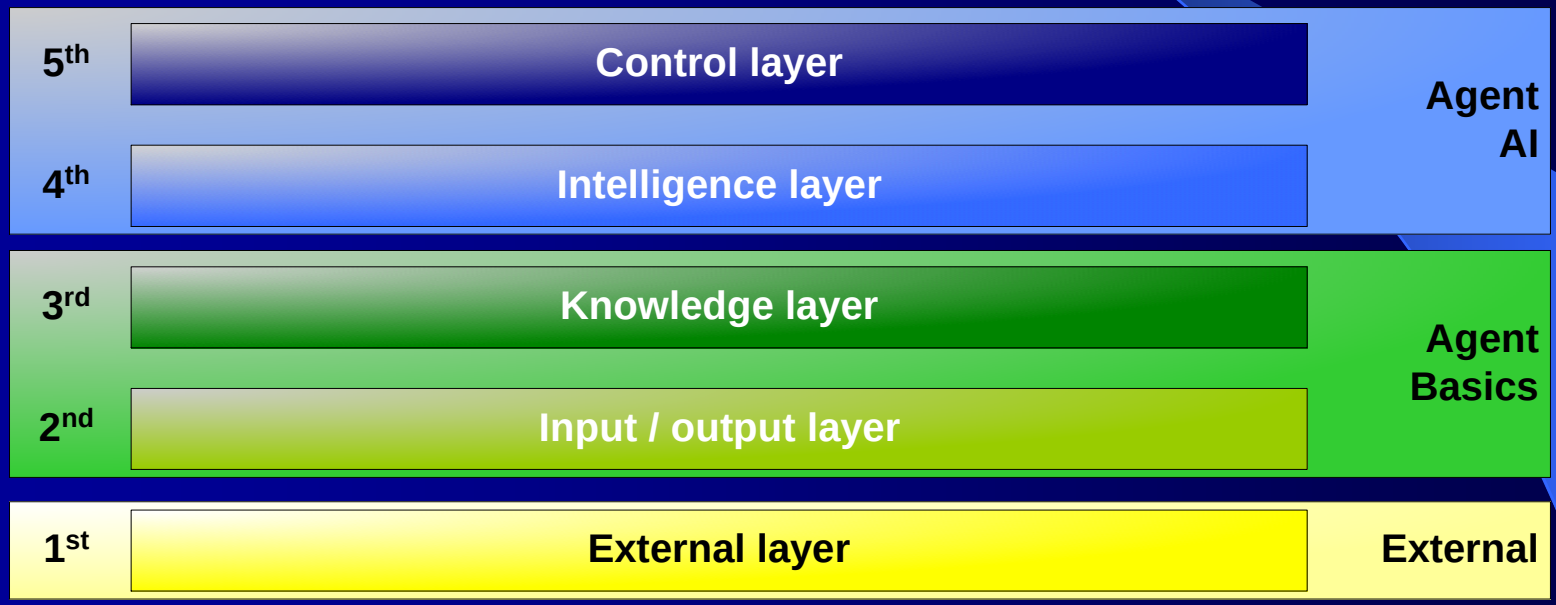
# System design



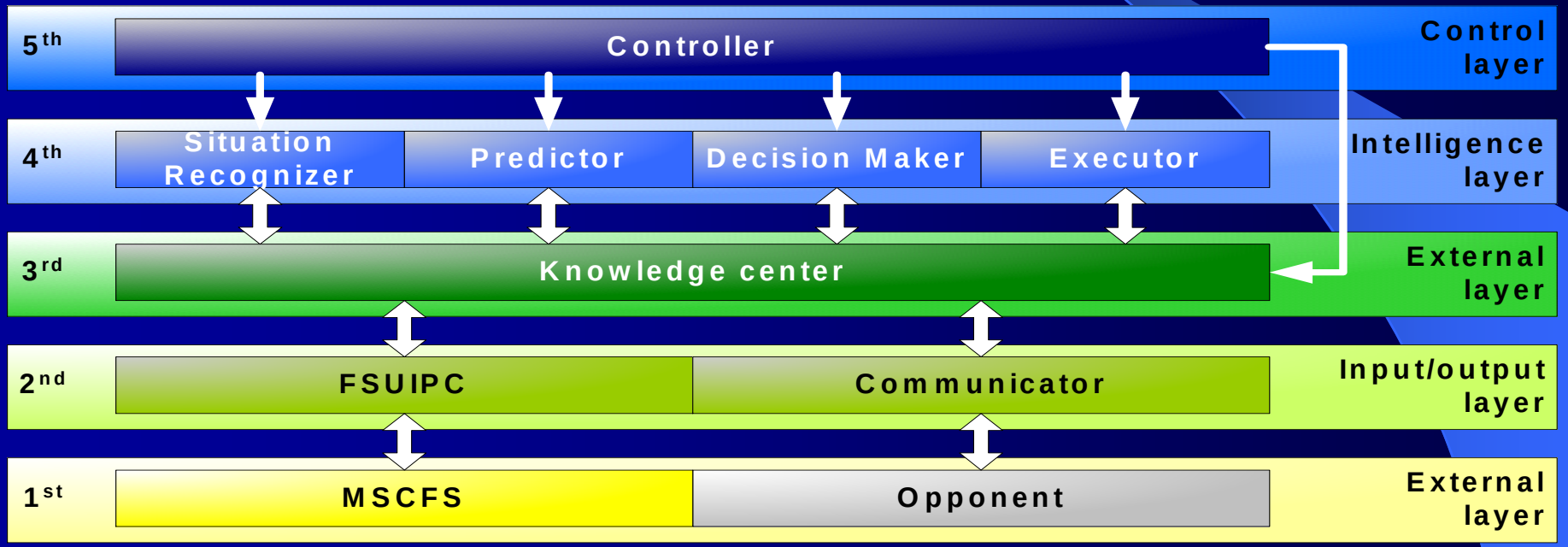
# System design



# Abstraction layers



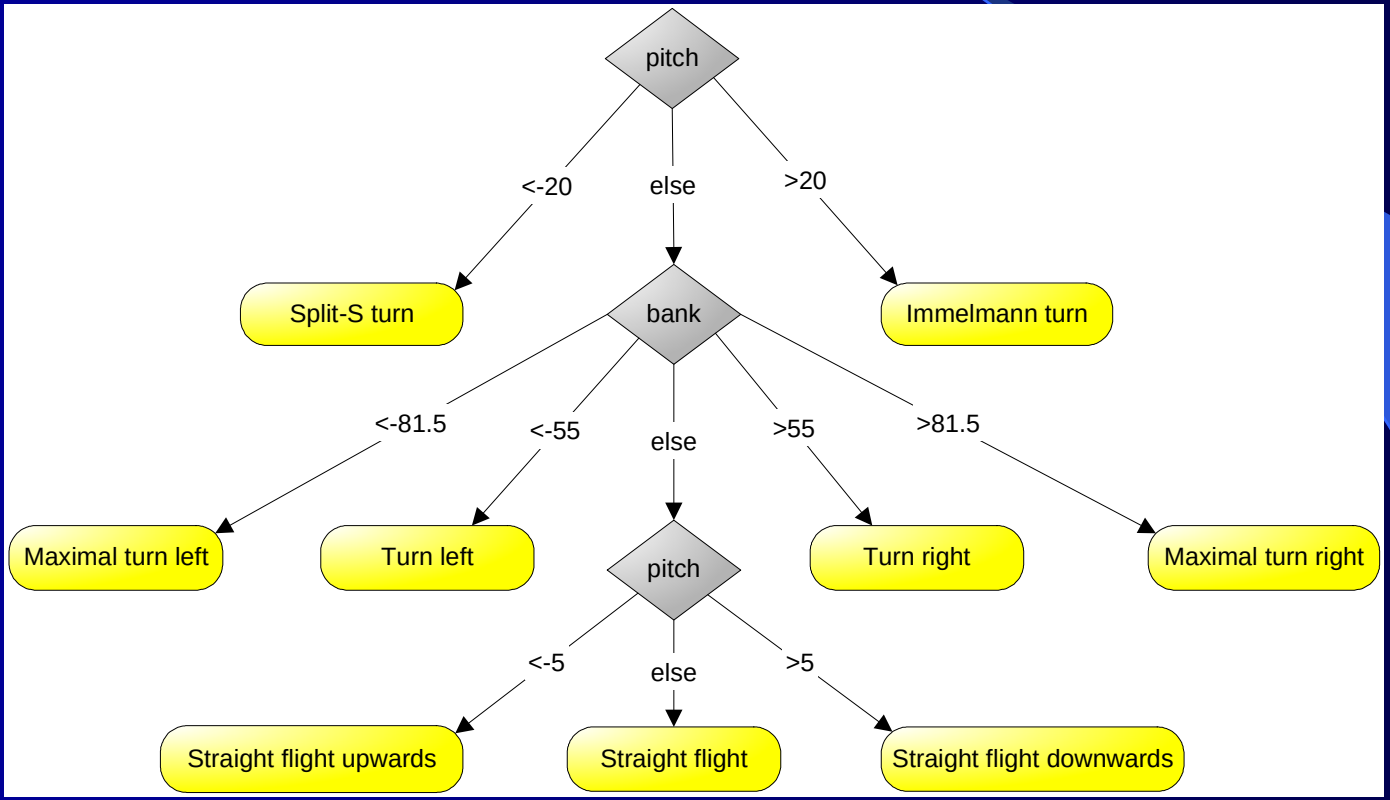
# Objects


5th
Controller
Control layer
4th
Situation Recognizer
Predictor
Decision Maker
Executor
Intelligence layer
3rd
Knowledge center
External layer
2nd
FSUIPC
Communicator
Input/output layer
1st
MSCFS
Opponent
External layer

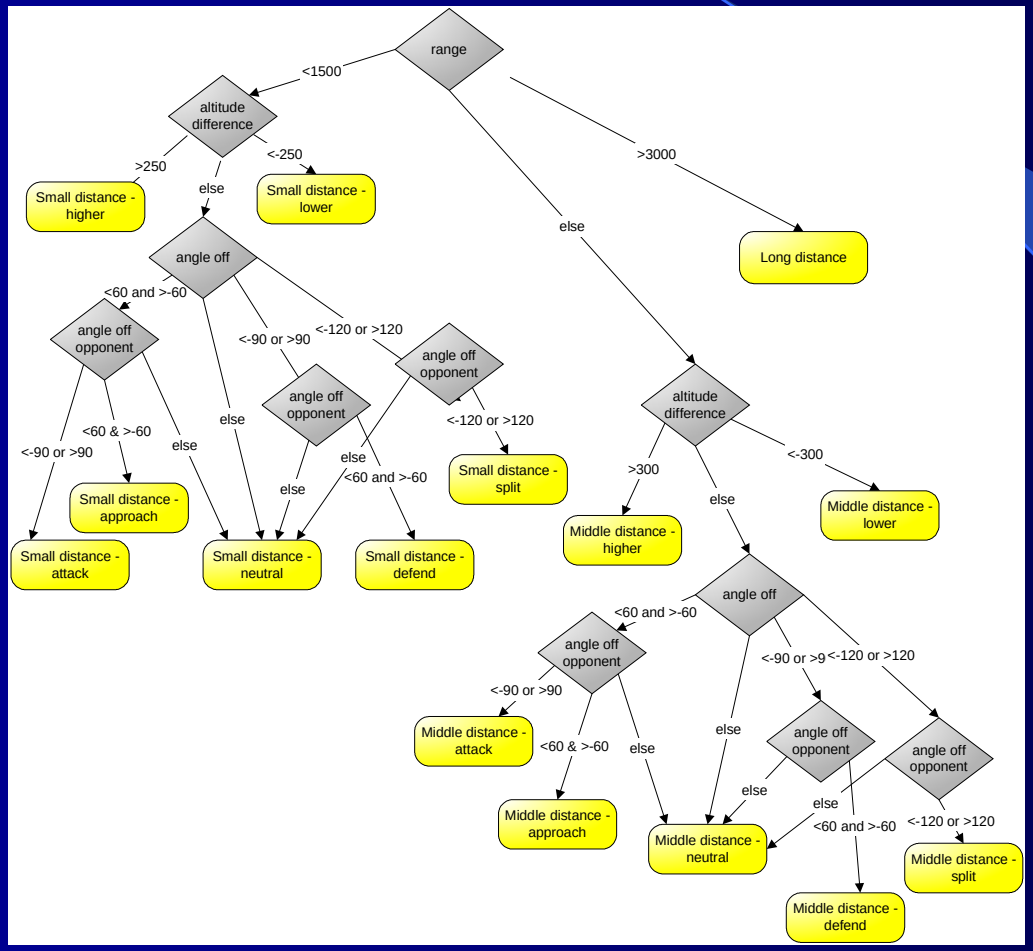
# A.I. objects

- Situation recognition (maneuver, position)
- Decision-making
- Maneuver execution

# Maneuver Recognition



# Position Recognition




introduction – **design** – results – conclusions

# Decision-making

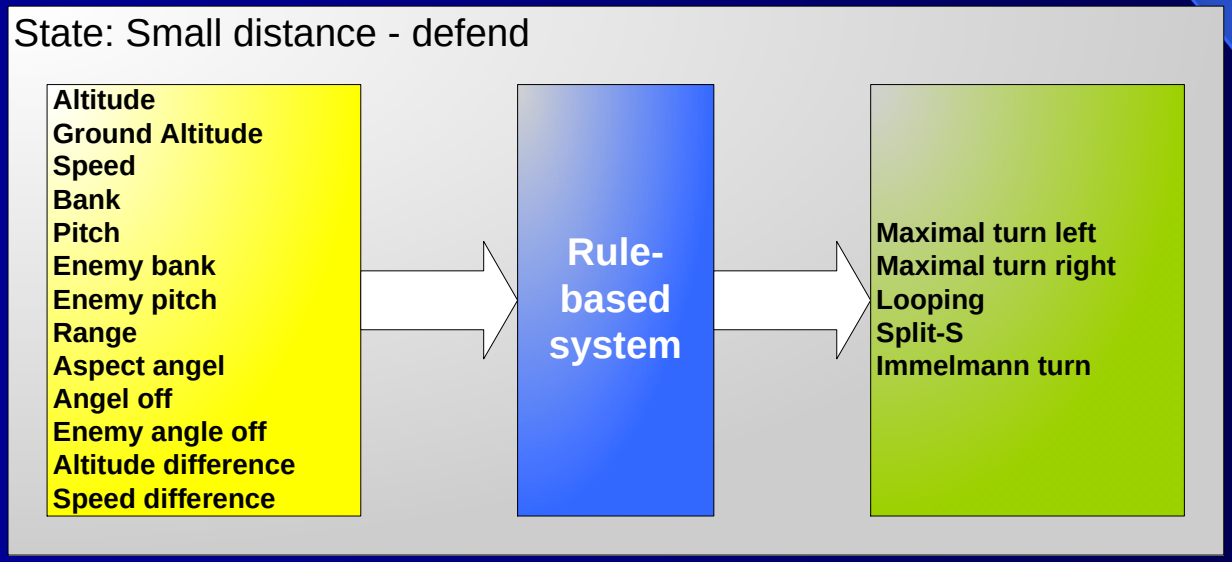
"IF angle off > 0 THEN turn right"

Acquisition of rules:

- From domain expert (pilot)
- Learn from observing domain expert
- Learn from experience

# State-based decisions

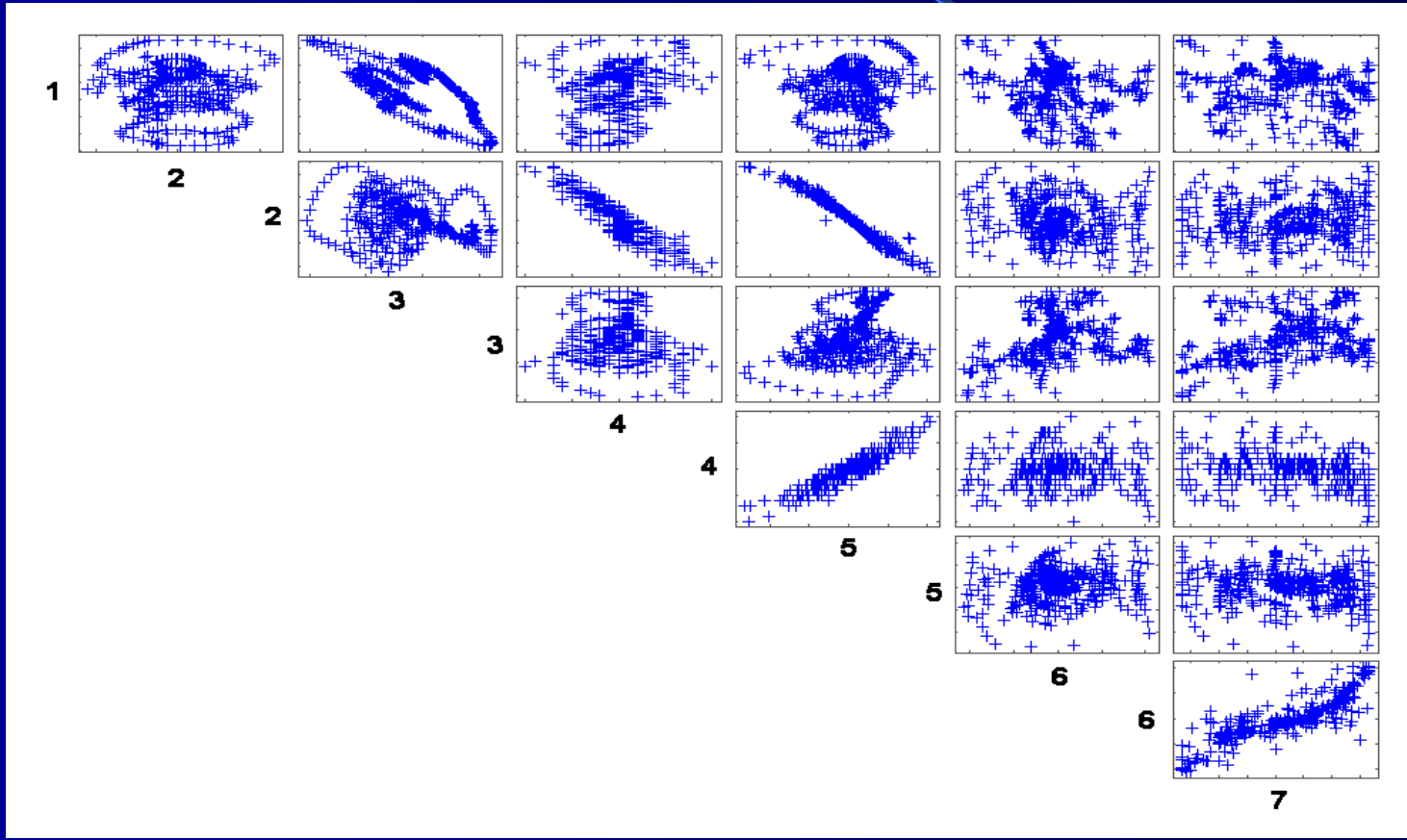
For each ‘position’ a separate set of rules



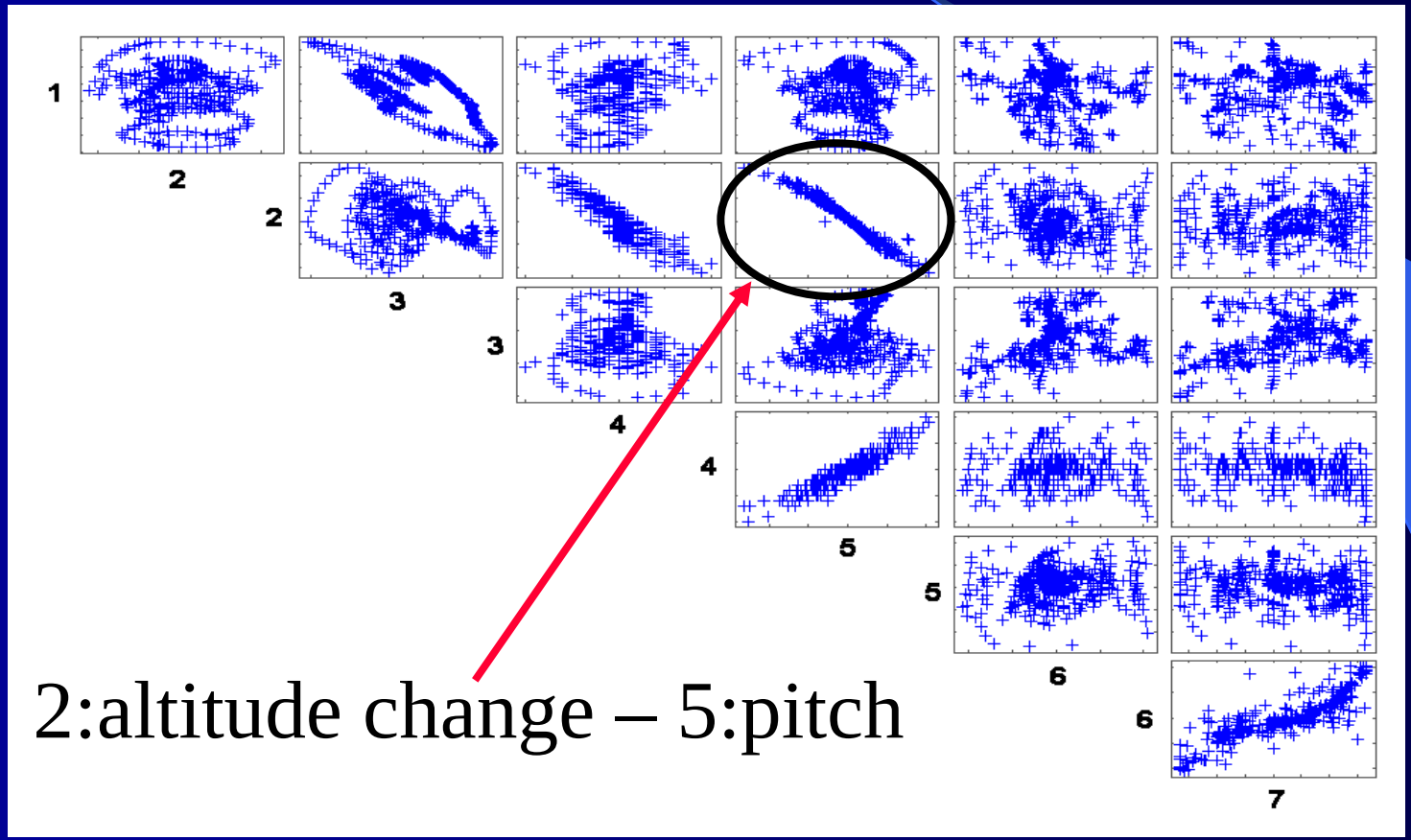
# Maneuver execution

- Find values for control parameters
- Approach of each maneuver is different
- Learn by observing experts (pilots)

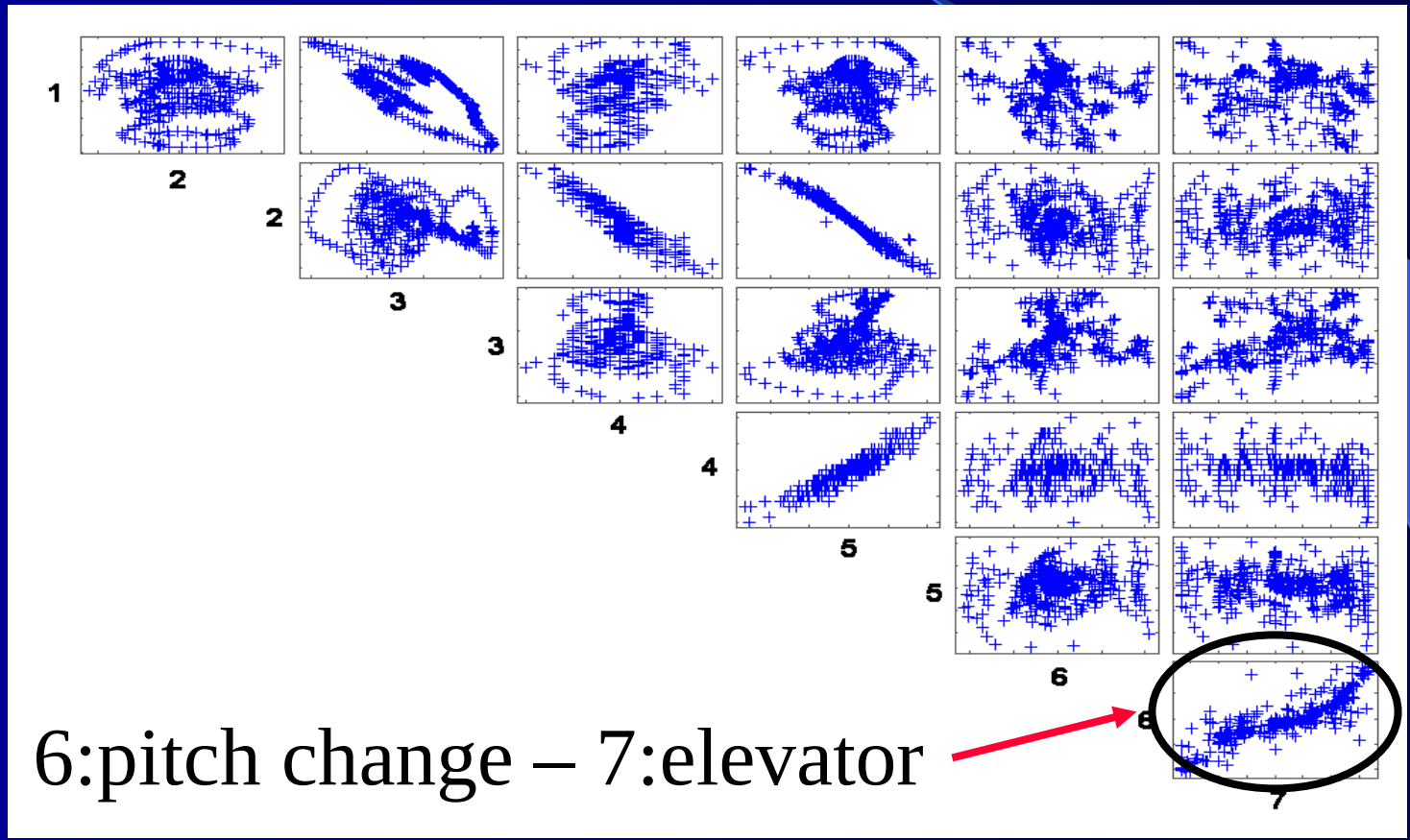
# Observing experts



# Observing experts



# Observing experts




6:pitch change – 7:elevator

# Regression

New value for ‘aileron’:

f(x = desired change ‘bank’) =
$1.115x^3 + 6.236x^2 - 1153x - 2745$

# Regression

Functions with 2 parameters:

- \+ Smoother maneuver execution
- \- Undesired behavior

# Example: step 1

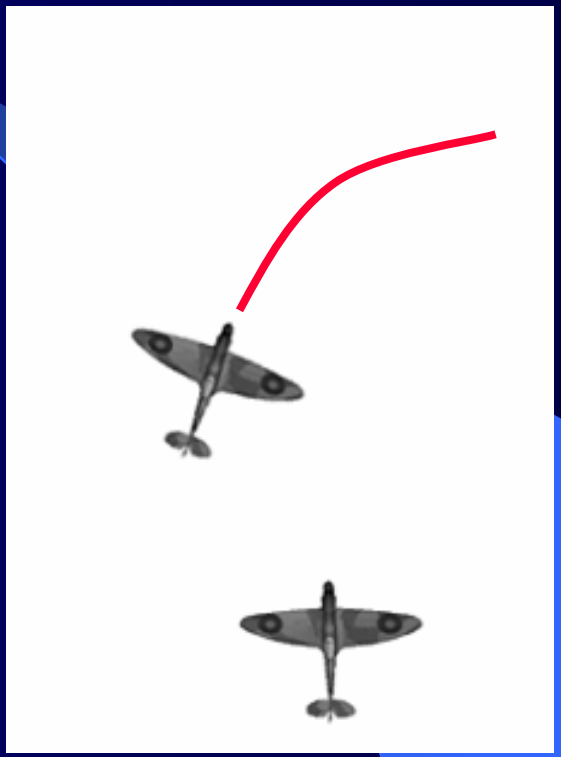
Retrieve data:

- Data from MSCFS and opponent
- Undependable and continuous process

# Example: step 2

Situation recognition:

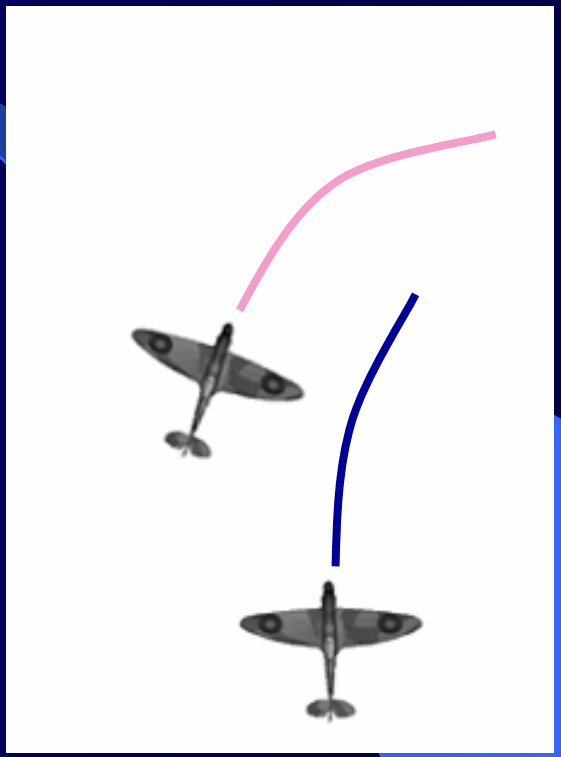
- Opponent maneuver: ‘turn right’
- Position: ‘small distance attack’



# Example: step 3

Decision-making:

- ‘angle off’ ( ao = -12)
- ‘aspect angle’ (aa = 15)
- maneuver ( om = ‘turn right’)

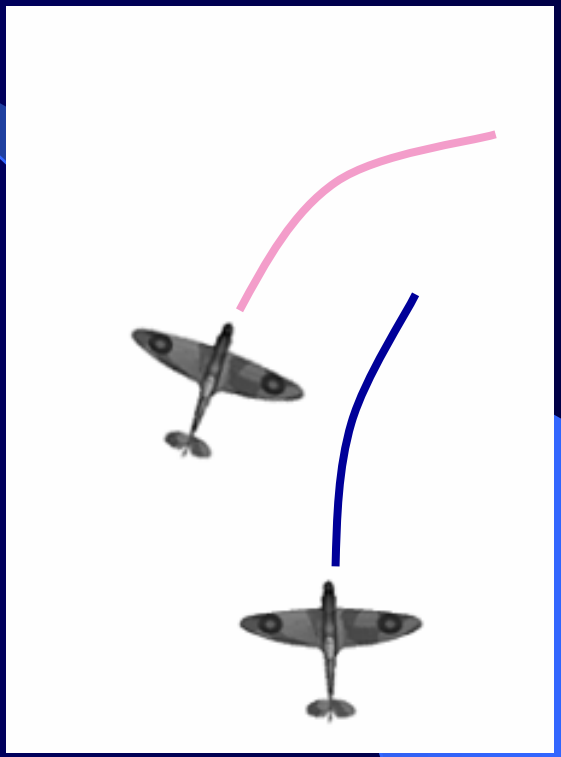


IF (ao < -5) AND (aa > 0) AND
(om == turn right) THEN ‘start a turn right’

# Example: step 4

Maneuver execution:

- aileron: 16000
- elevator: 12000



# Example: step 5

Further:

- Wait
- Start reasoning again
- 3 times a second

# Implementation

Prototype in C++:

```
        // Compute difference between current bnk and desired bnk
        // Desired bank is 2/3 * 85 degrees + 1/3 * desired bank
        diff = max(-10,min(10,(((desbnk+dir*2*85)/3) - data->bnk)));

        // Compute the ail from the difference in desired bnk and current bnk
        if (dir > 0)
            data->ail = (short)(max(-16000,min(16000,(1.115 * pow(diff,3) + 6.236 * pow
        else
            data->ail = (short)(max(-16000,min(16000,(1.748 * pow(diff,3) -12.269 * pow
    }
```

# Implementation

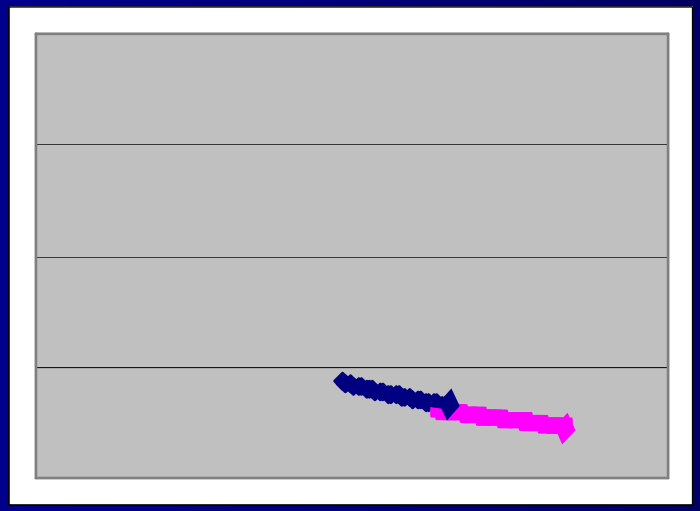
Limitations of the implementation:

- 1 type of airplane (Spitfire)
- Always ‘full throttle’
- No shooting

# Results

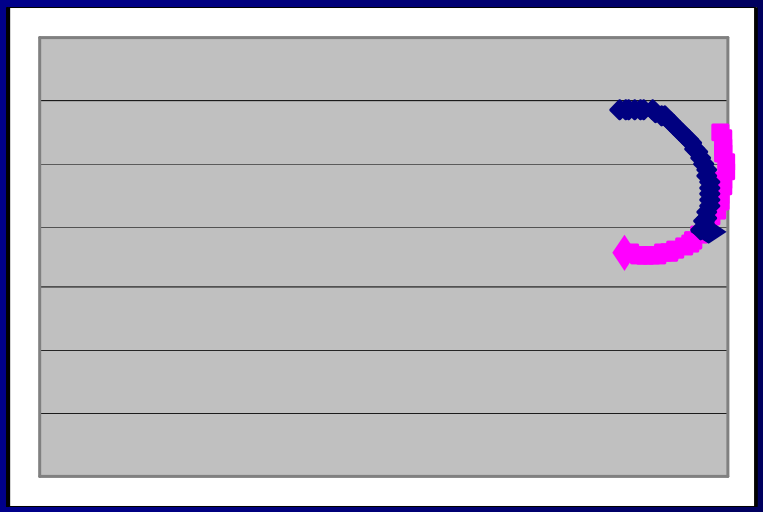
# Scenario 1

Straight flight



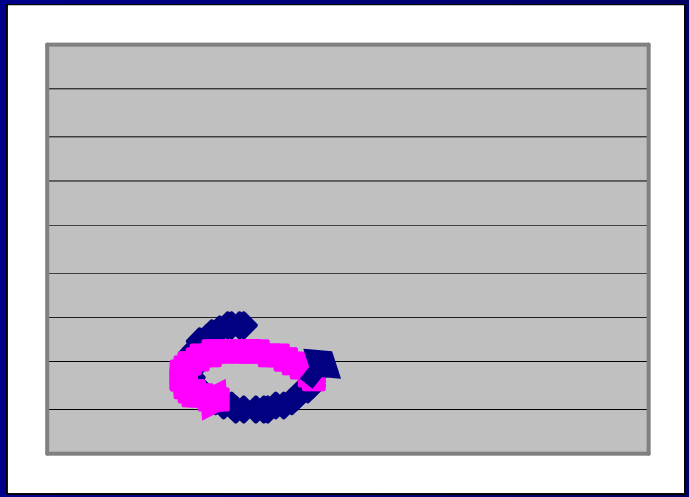
# Scenario 2

Simple turns



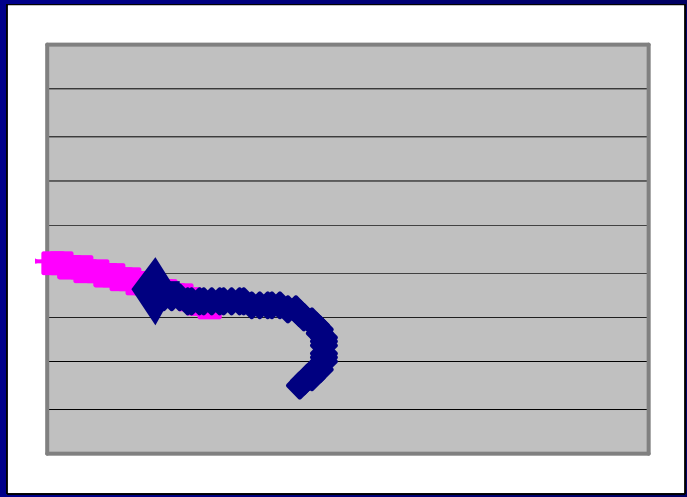
# Scenario 3

Dogfight

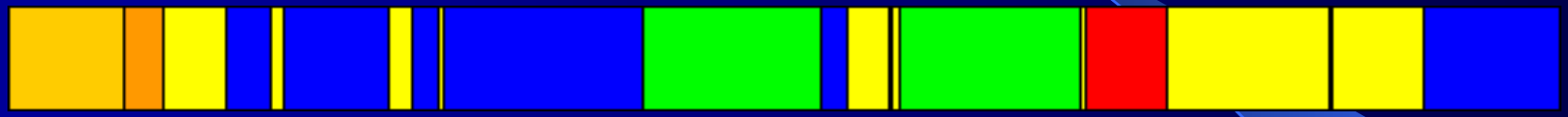


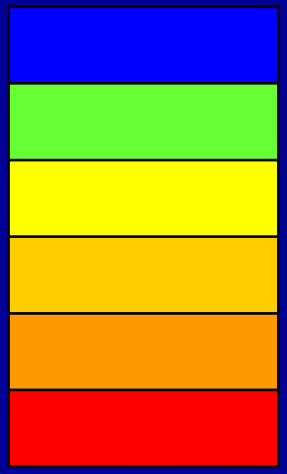
# Scenario 3

Dogfight



# Scenario 3: positions

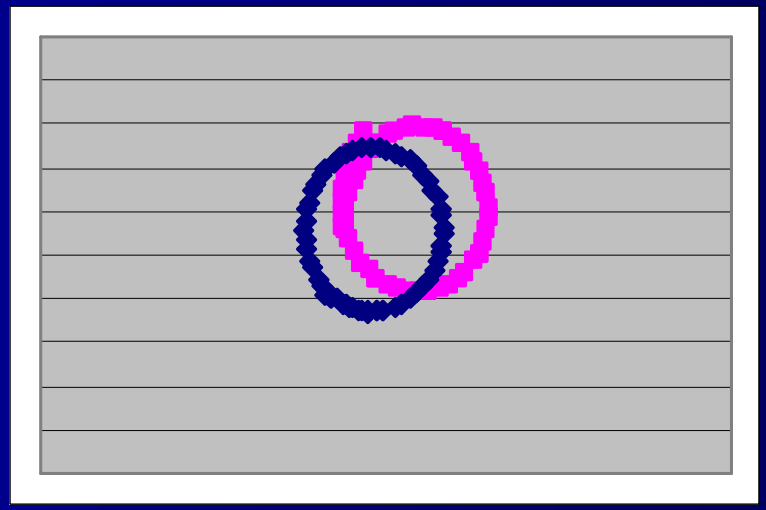




short-range-attack
short-range-higher
short-range-neutral
middle-range-approach
short-range-approach
short-range-defend

# Scenario 4

Agent - agent



# Demo

# Conclusions

# Project goal

Create an agent that can play a dogfight in MSCFS

# Conclusions

- Agent reacts faster than human
- Agent is less creative as human
- All tasks are important

- Agent works succesfully!

# Future work

- Extends and improve decision-making rules
- Add prediction
- Decision-making different per position
- Adaptive flight behavior
- Probabilistische approach

# Game Over