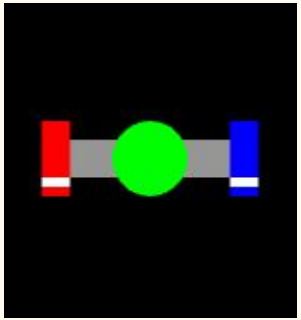
EA Drone Controller (*Good boy drone*)

Marek Bečvář

Background

- Custom CPP implementation

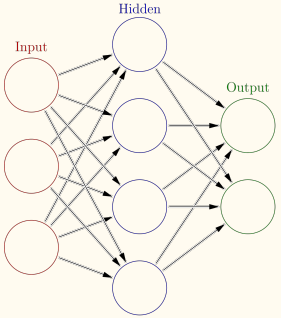
- Custom drone structure with 2 independent engines



- SFML cross-platform graphical interface

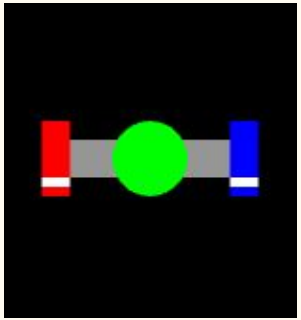
- Custom simplified physics solver

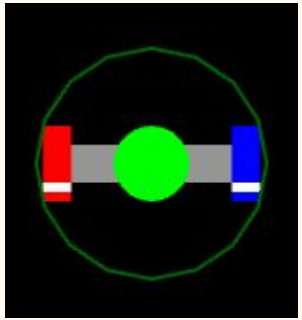
- Custom Neural nets for drone brain

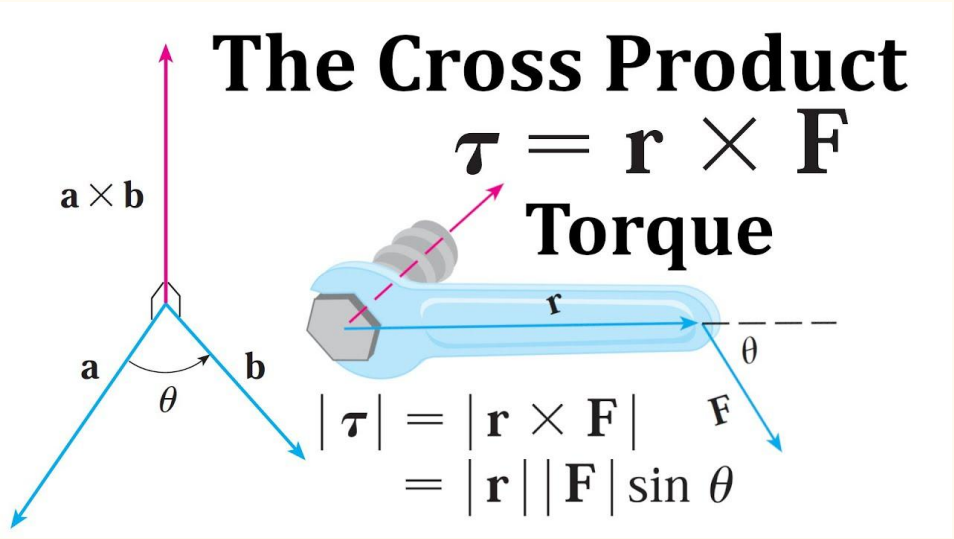


Physics

1. Building custom physics solver - no real reason but unified the process
2. NOT difficult – Gravity & Torque
3. Allow for adjustable engine angles
4. Basic collision handling

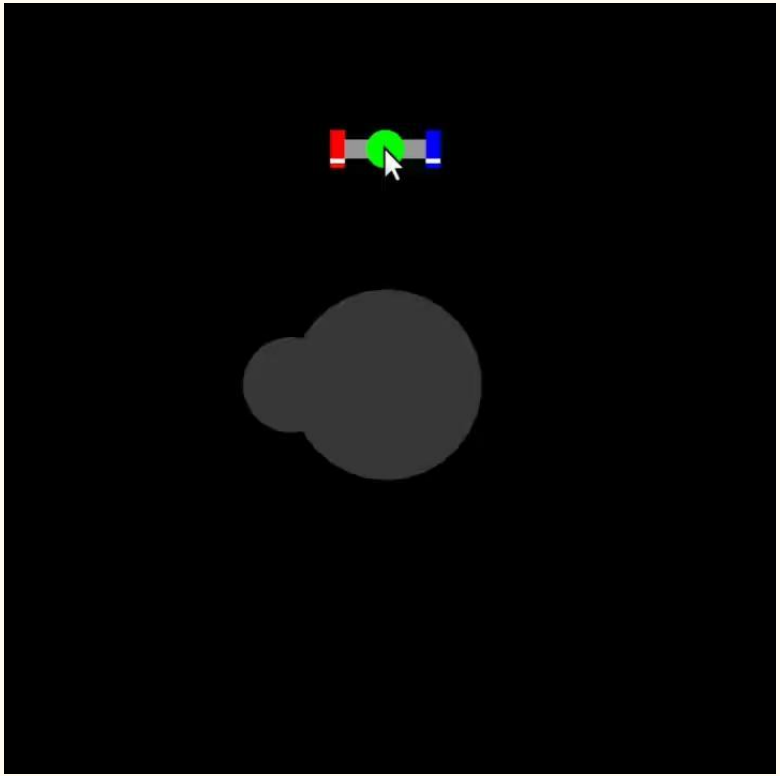







```
float getTorque() {
    const float lPower = thrusterLeft.power;
    const float rPower = thrusterRight.power;

    const float lAngle = thrusterLeft.angle - M_PI * 0.5f;
    const float rAngle = thrusterRight.angle - M_PI * 0.5f;
    sf::Vector2f lThrustVector{cos(lAngle), sin(lAngle)};
    sf::Vector2f rThrustVector{cos(rAngle), sin(rAngle)};

    // works for 2 symetrical thrusters
    float lTorque = cross(lPower*lThrustVector, -thrusterOffset);
    float rTorque = cross(rPower*rThrustVector,  thrusterOffset);

    // not really physicy but helps even out the cross product value
    const float momentOfInertia = 0.0005f;

    return (lTorque + rTorque)*momentOfInertia;
}
```

Physics Collision detection



Evolutionary Algorithms – Inputs/Outputs/Fitness

- Agent Inputs (7/8):
  - 2x velocity
  - angular velocity
  - cos & sin of drone angle
  - x & y distance to goal
  - (raycasting directional sensor)

- Outputs (4):
  - Power for left & right motor
  - Angle for left & right motor (gradual application)

- Fitness:
  - **Running fitness:** cos^4 of goal distance added at every step multiplied by *current goal index*
  - **Goal collected:** *Alive timer * current goal index*

- After collecting goal – *Alive timer* divided by 0.5

Evolutionary Algorithms

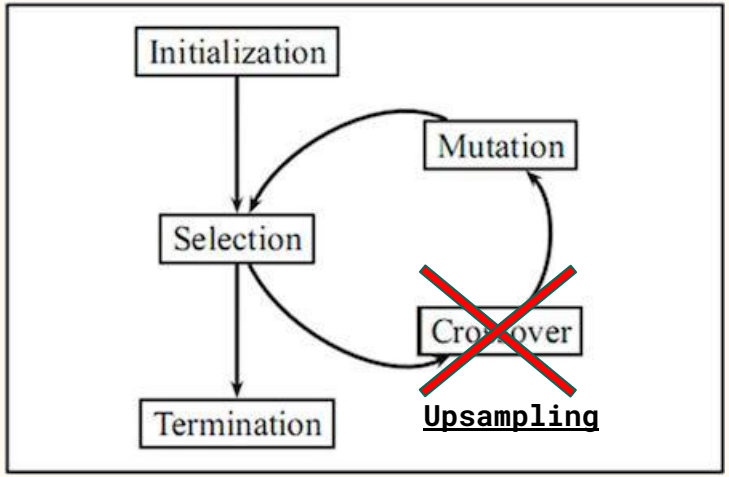
1. **Basic EA** with tweaks – Crossover doesn't make sense for NNs

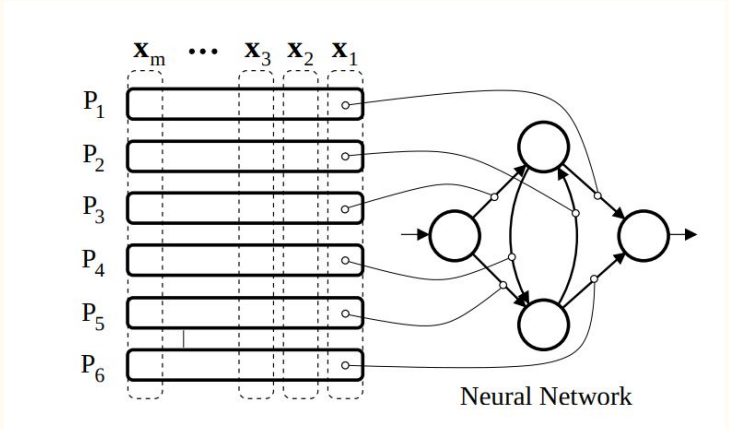
   Top-k (10-20%) selected – cloned and mutated into the rest of the population, diversity falls off

2. **CoSyNE** – Algorithm from the lecture

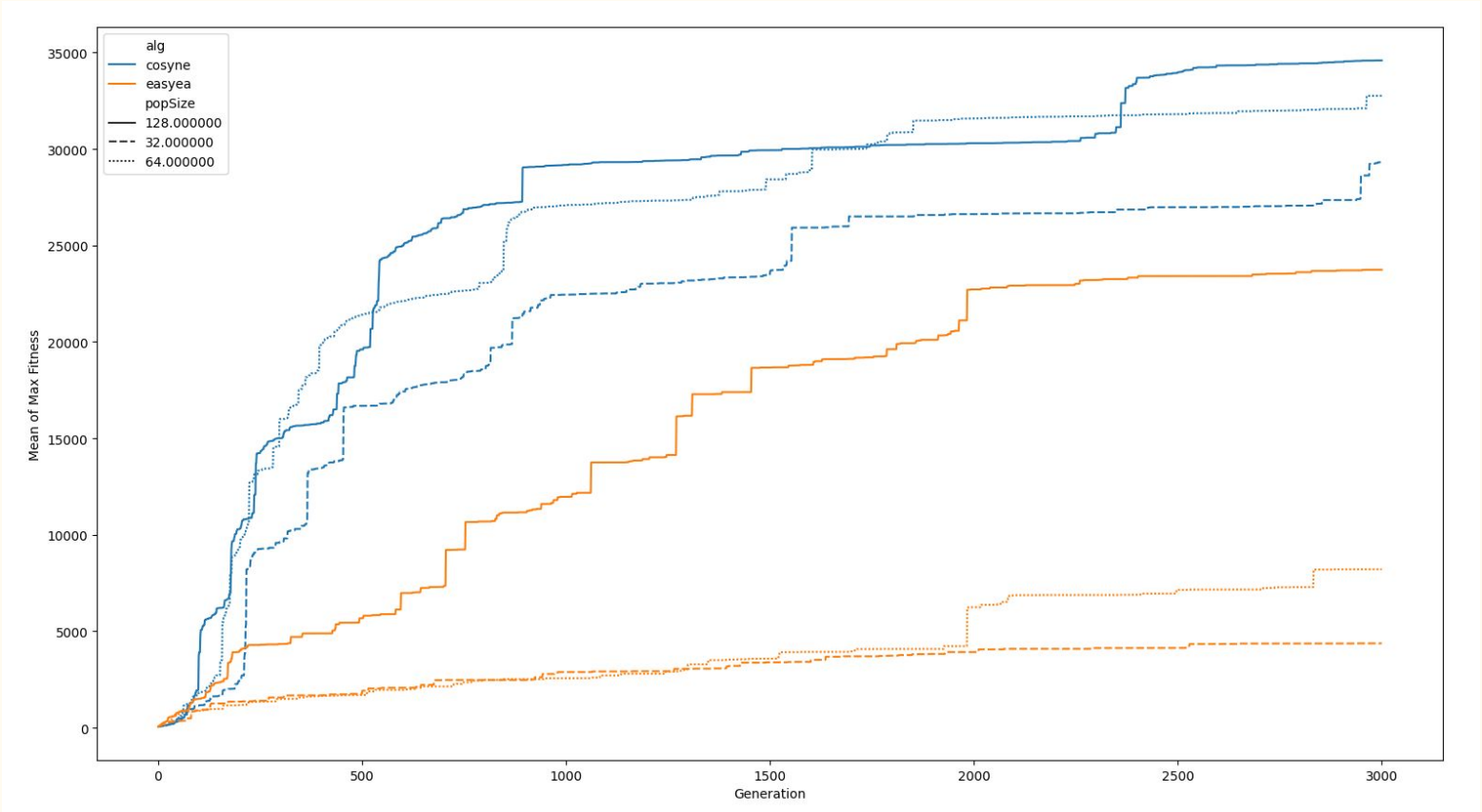
   Each weight developed in a subpopulation of weights – higher diversity – longer improvements

   Always using incremental evolution

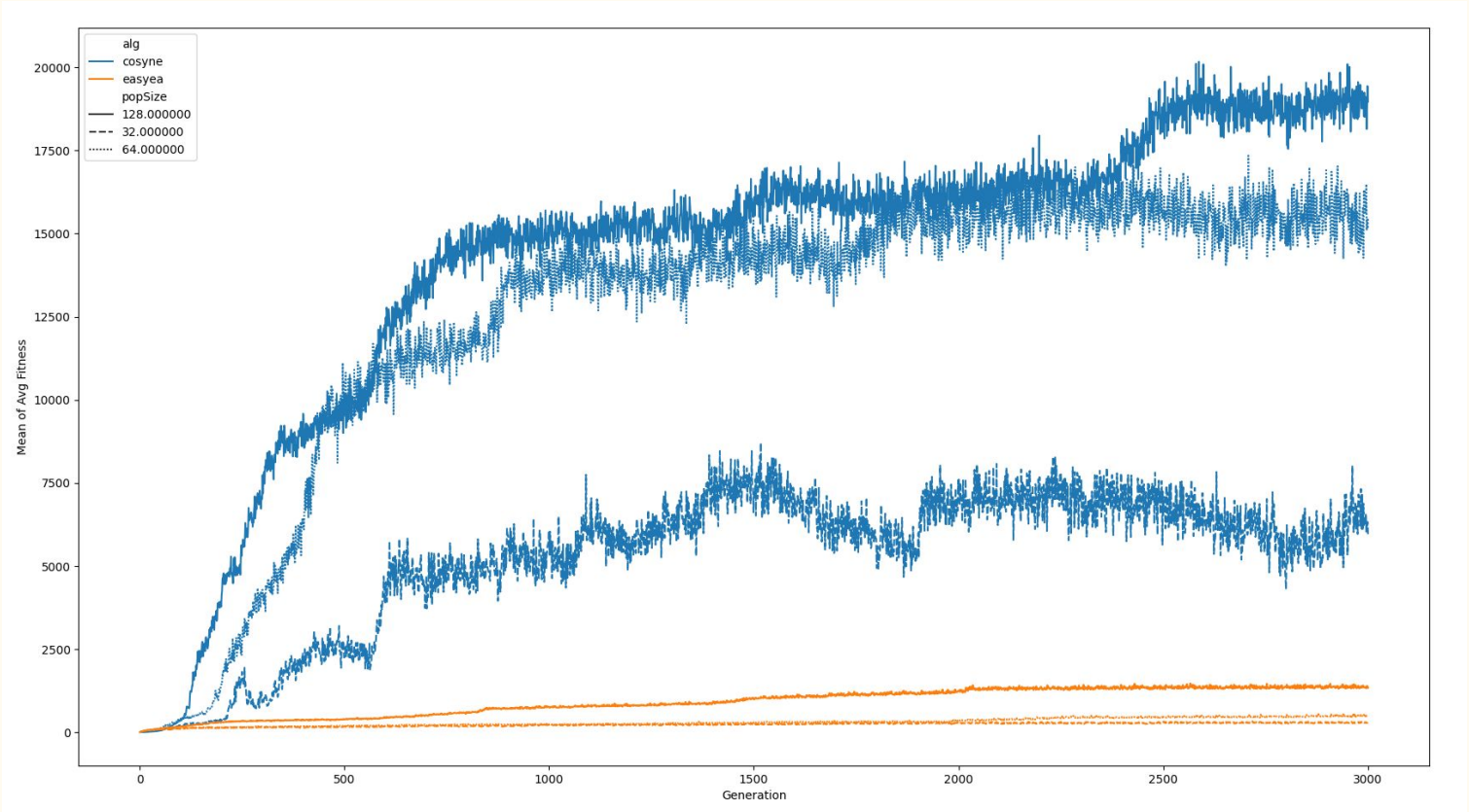




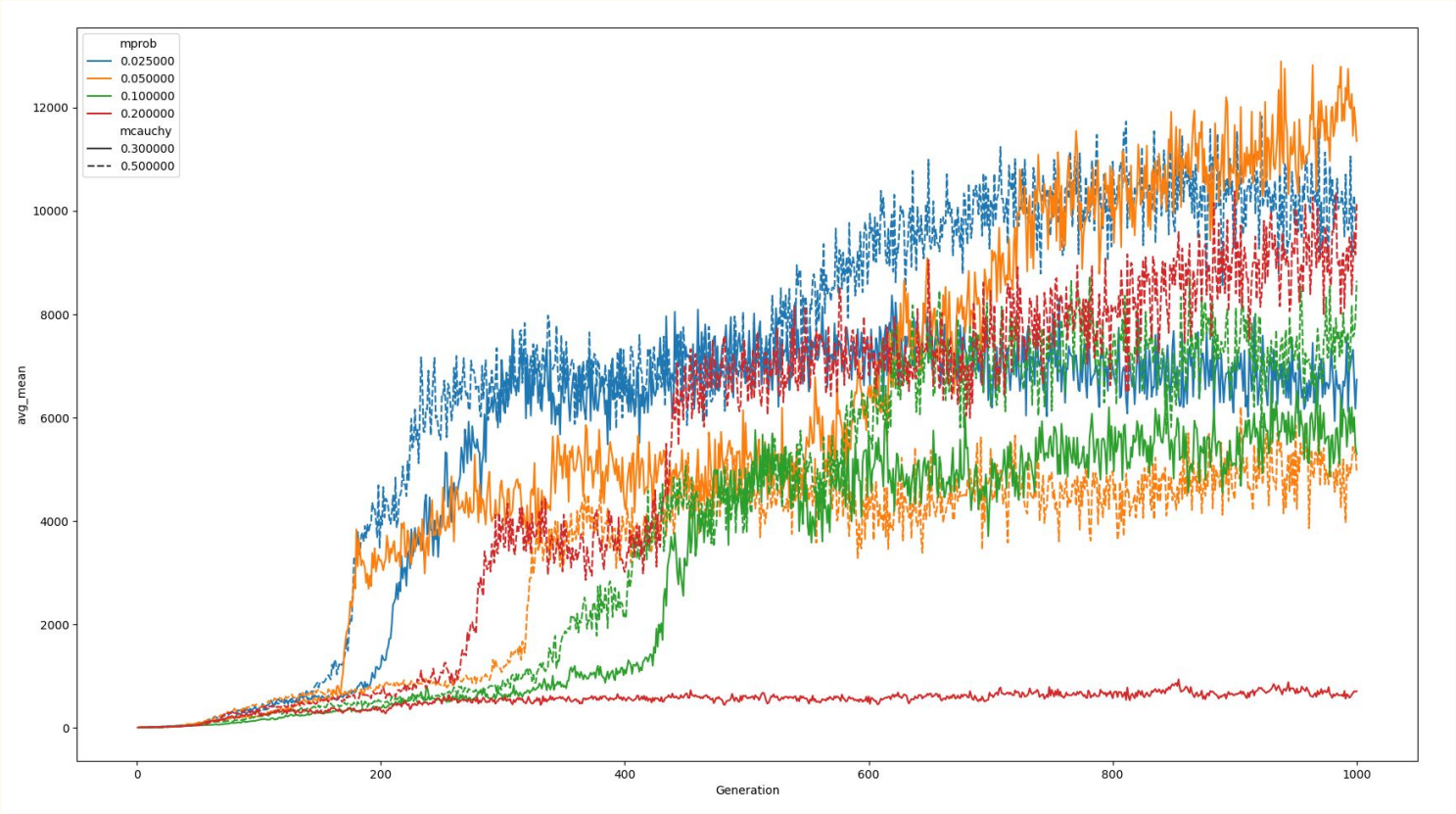
Evolutionary Algorithms – Comparison



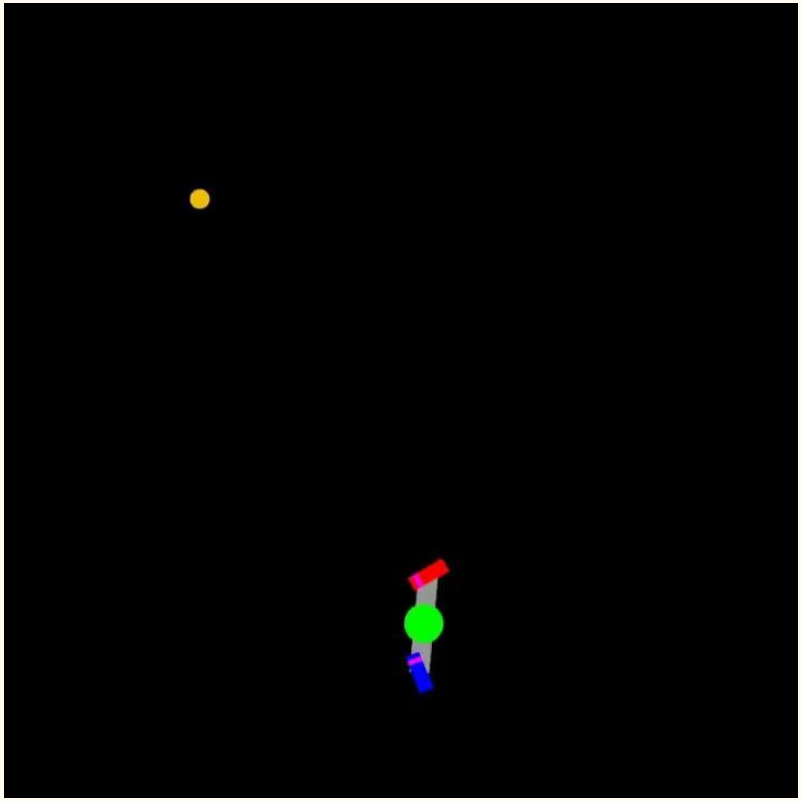
Evolutionary Algorithms – Comparison

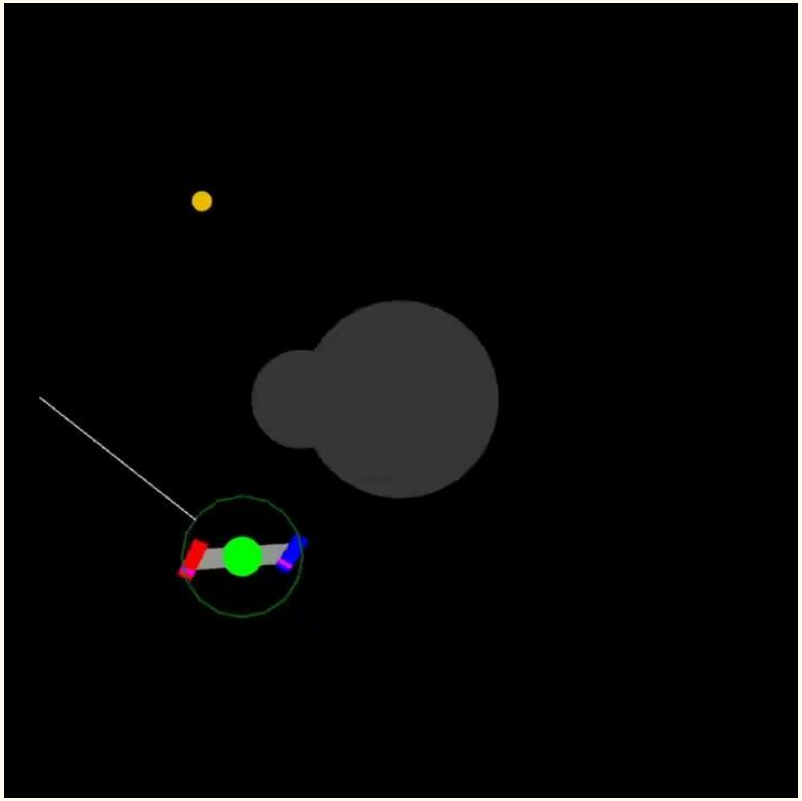


CoSyNE – grid search for mutation params



Results – Incremental Evolution





Results – Human Controllable

Conclusion

The project showed the difficulty of evolving NNs and strengths of advanced algorithms like CoSyNE.

Future works...

- Test more algorithms (NEAT)
- New levels for the incremental evolution
- Drone swarm/Controlled drone swarm
- 3D (Unity Game engine)?

Thank you for your attention
—