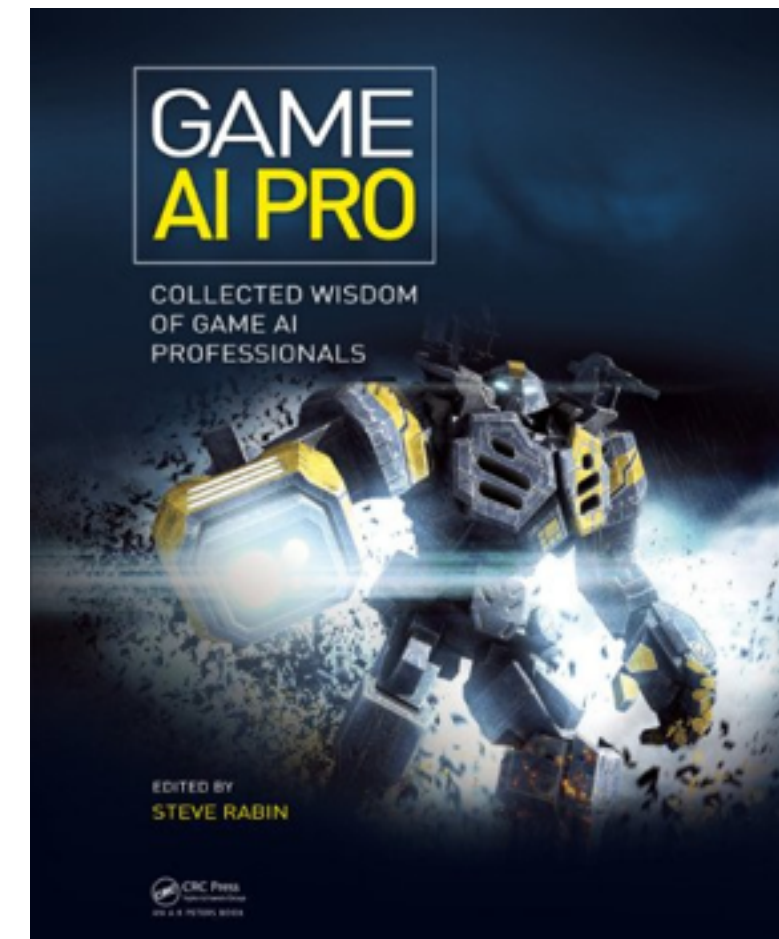


Crowd Simulation through Steering Behaviors and Flow Fields

Graham Pentheny

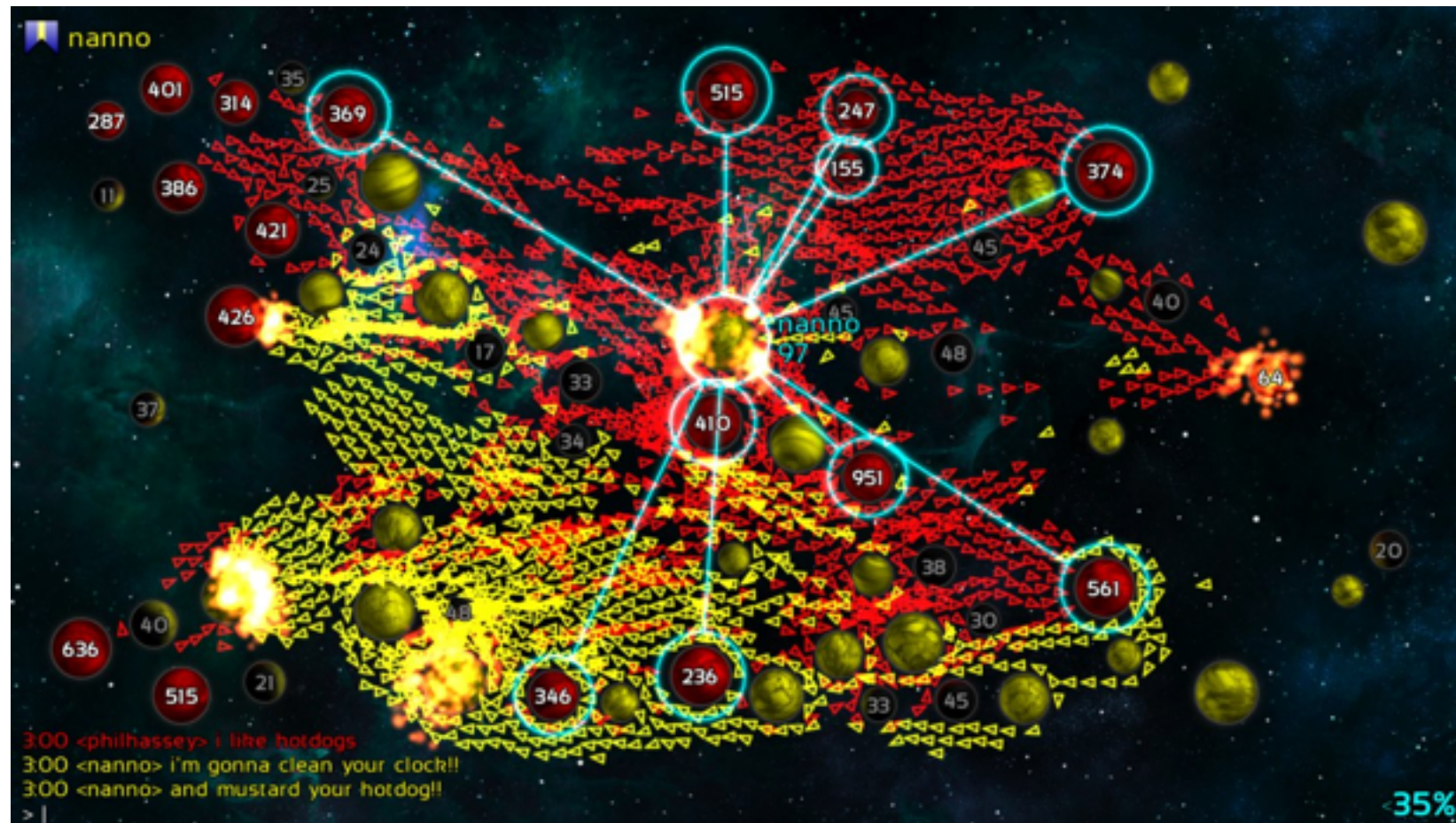
Independent Game Developer & AI Researcher

Graham Pentheny



“Effective Crowd Simulation
for Mobile Games”

Steering



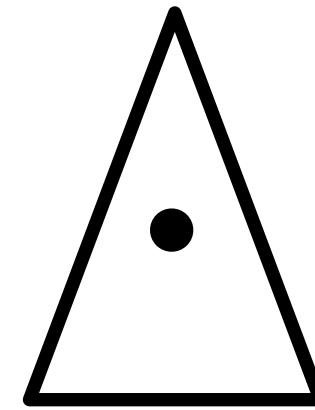
Galcon Fusion,
Copyright © 2012 Phil Hassey.



StarCraft II: Heart of the Swarm,
Copyright © 2013 Blizzard
Entertainment, Inc.

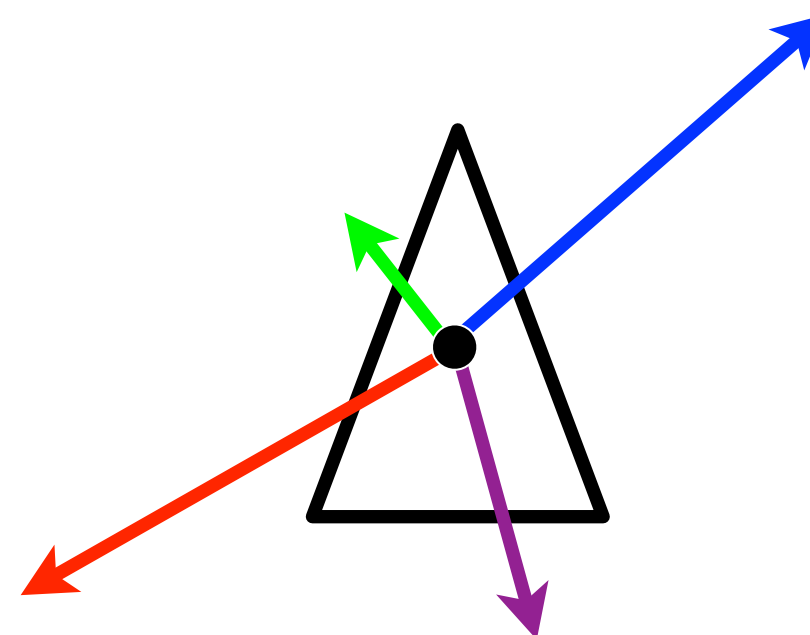
Steering

- Component system
- Specialized Behaviors
 - Encapsulate separate concerns
- Arbitration Function
 - Combine behaviors intelligently



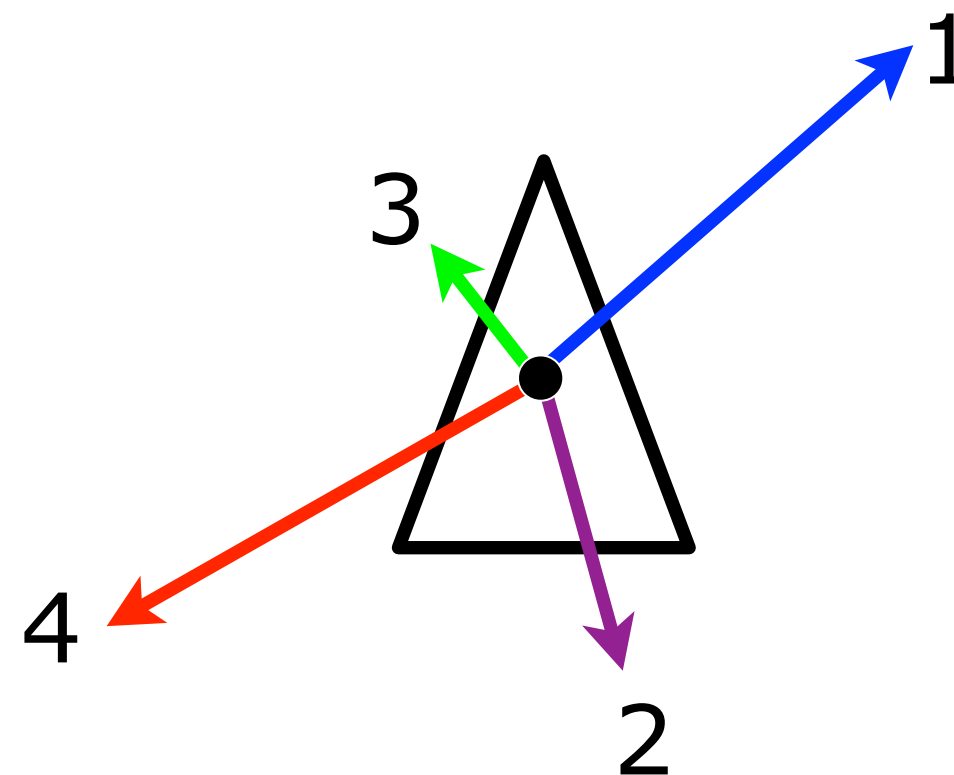
Steering

- Component system
- Specialized Behaviors
 - Encapsulate separate concerns
- Arbitration Function
 - Combine behaviors intelligently



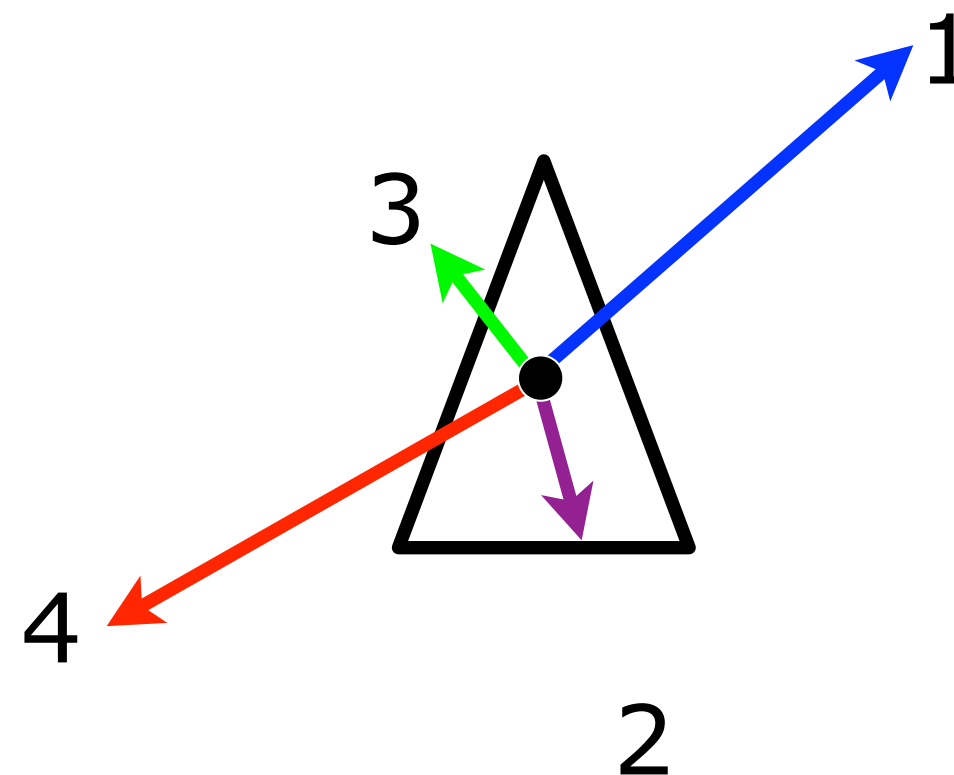
Steering

- Component system
- Specialized Behaviors
 - Encapsulate separate concerns
- Arbitration Function
 - Combine behaviors intelligently



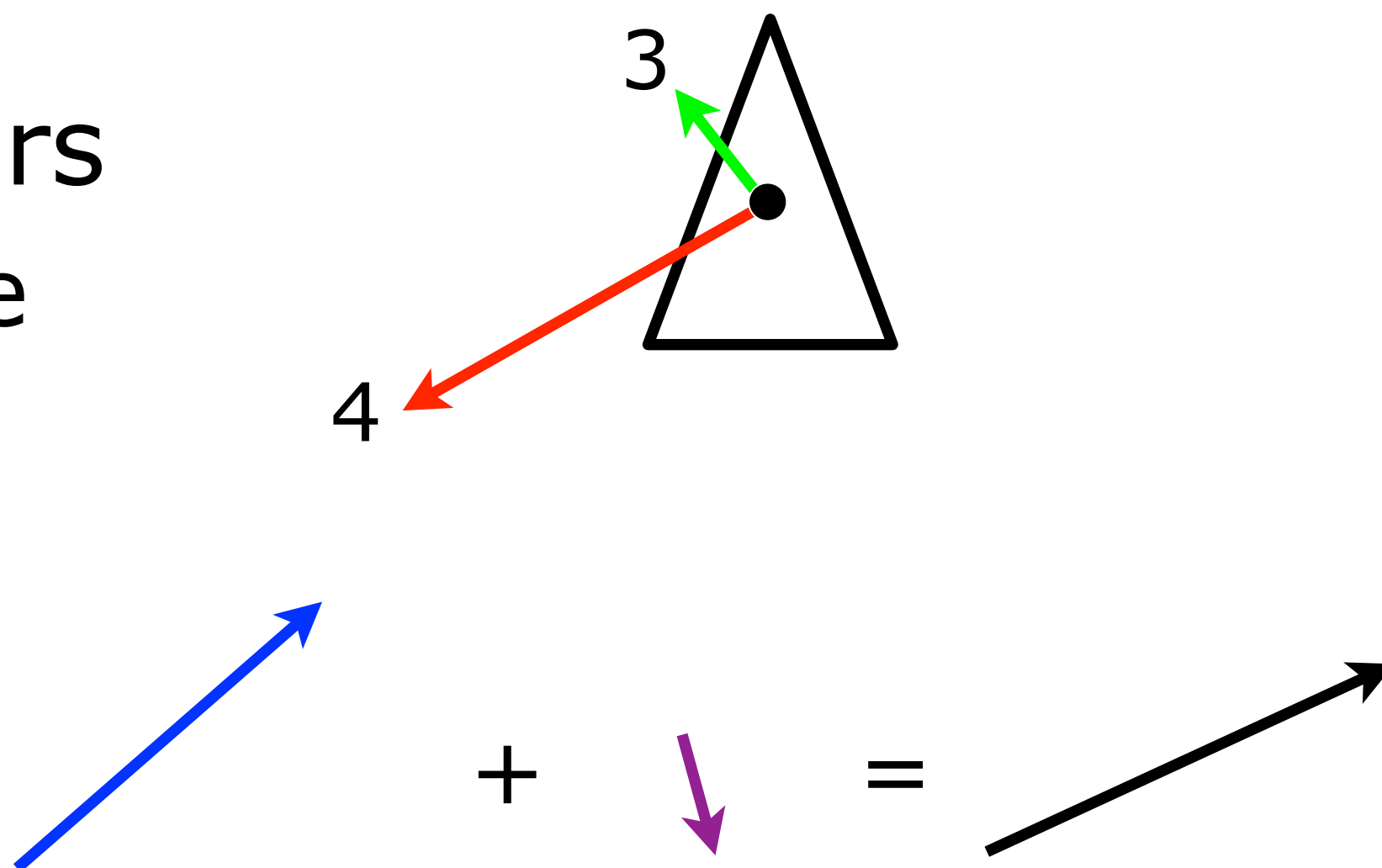
Steering

- Component system
- Specialized Behaviors
 - Encapsulate separate concerns
- Arbitration Function
 - Combine behaviors intelligently



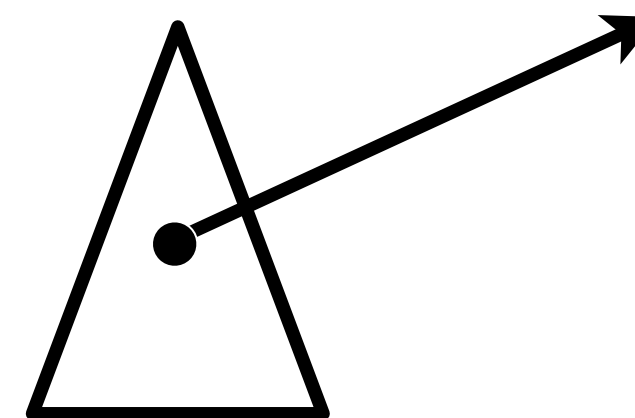
Steering

- Component system
- Specialized Behaviors
 - Encapsulate separate concerns
- Arbitration Function
 - Combine behaviors intelligently



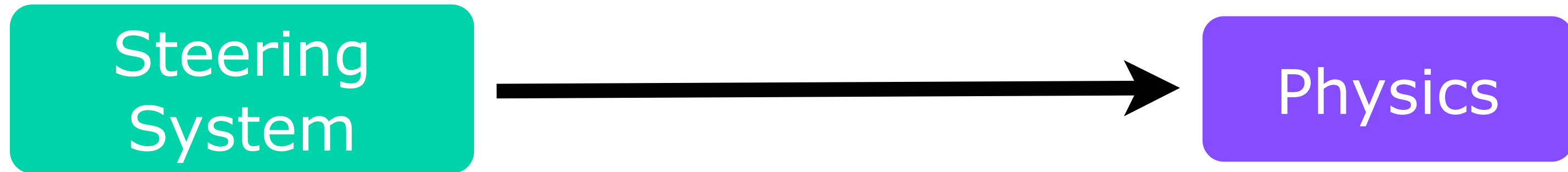
Steering

- Component system
- Specialized Behaviors
 - Encapsulate separate concerns
- Arbitration Function
 - Combine behaviors intelligently

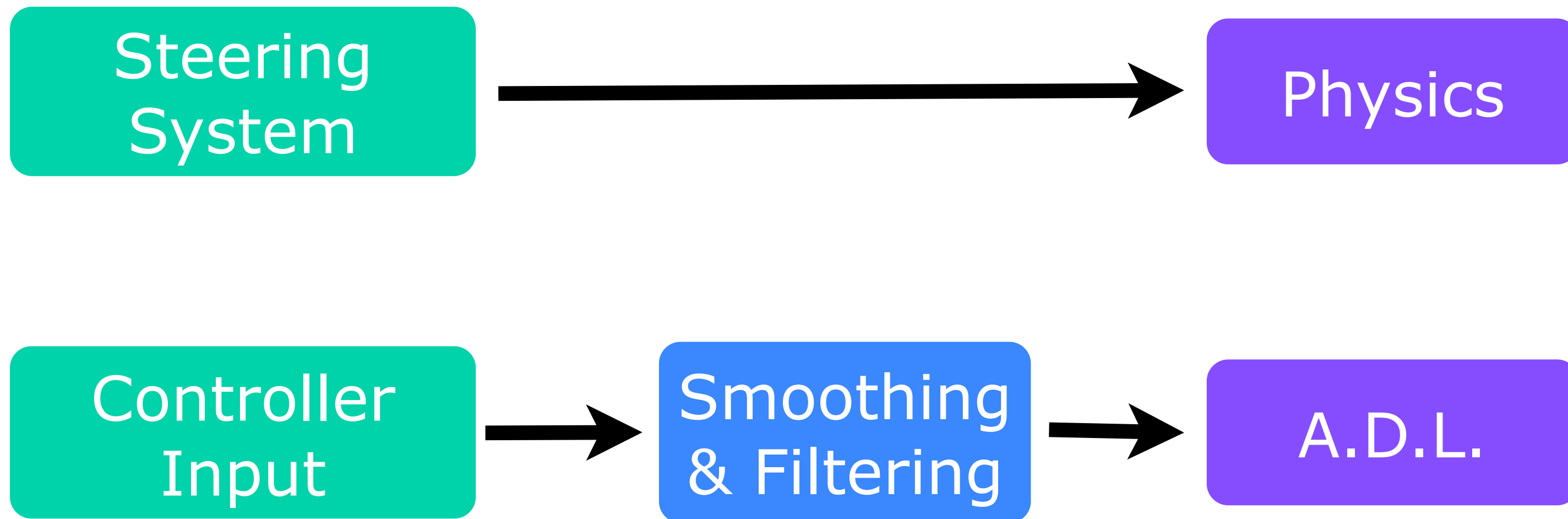


Steering

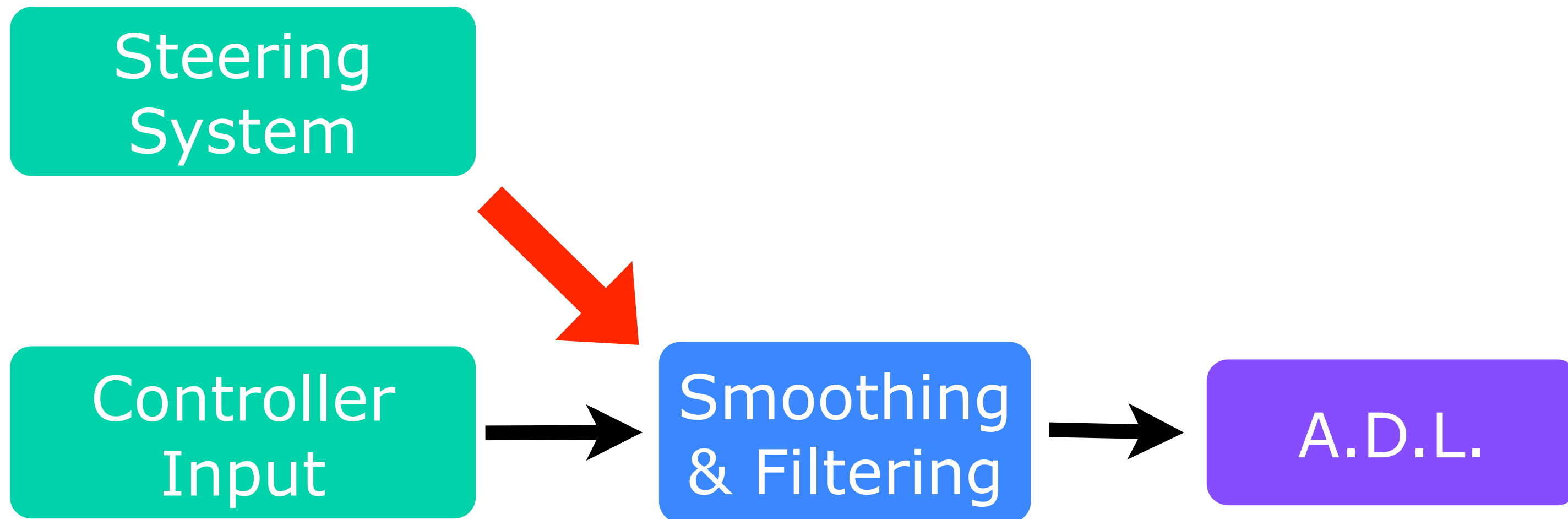
Steering



Steering



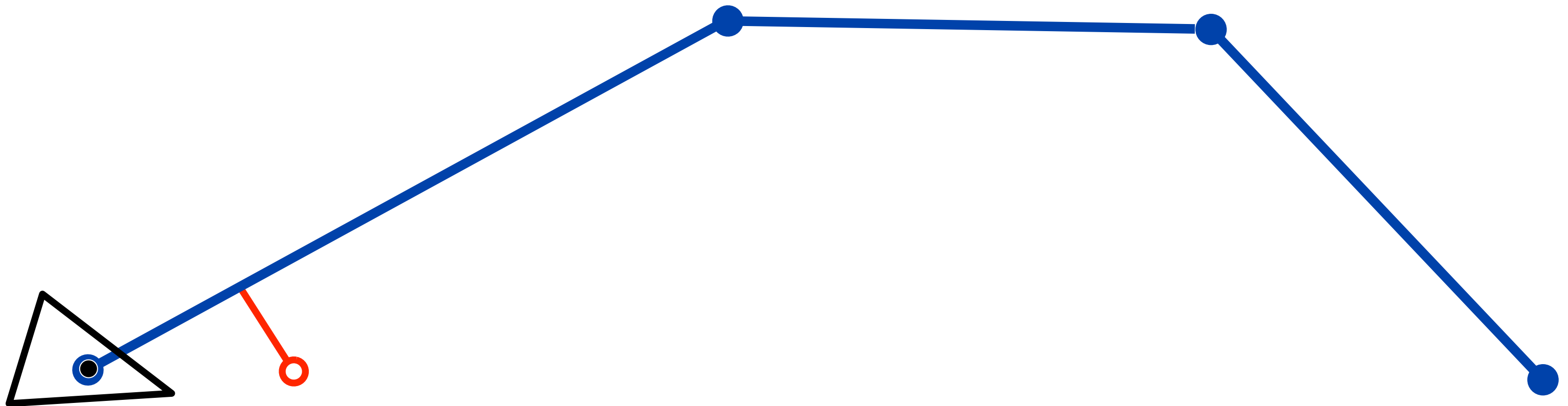
Steering



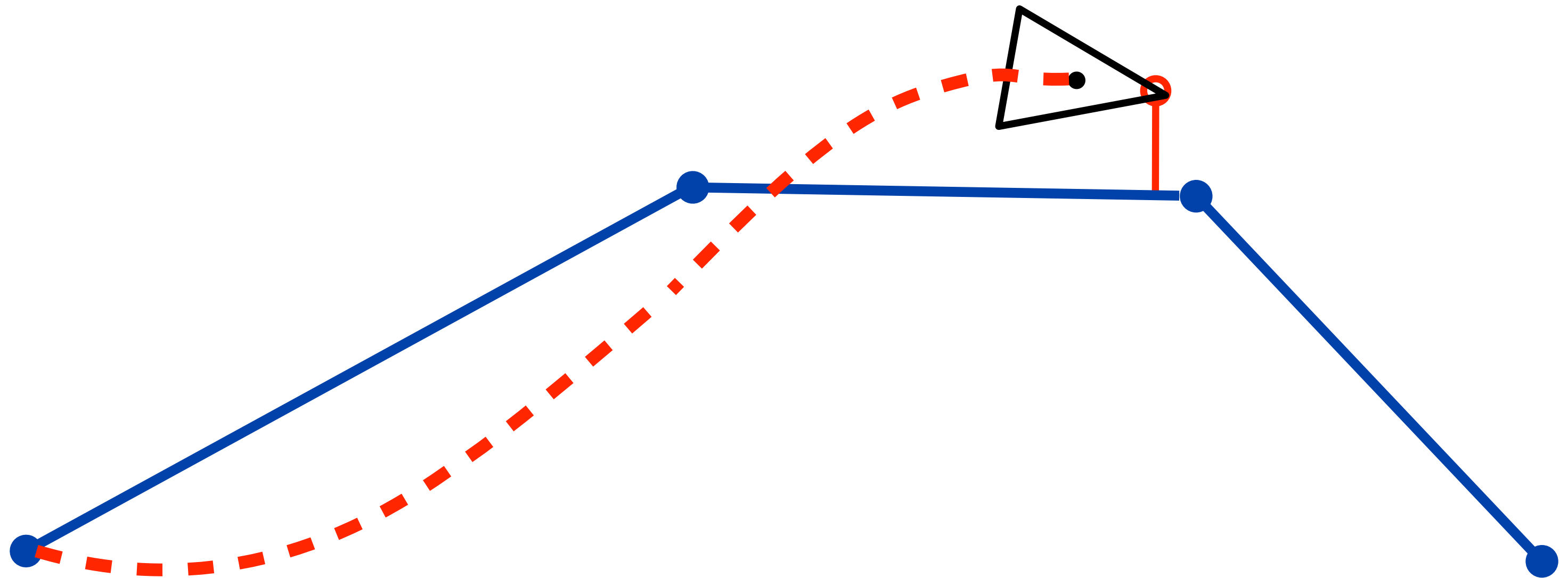
Traditional Crowd Simulation

- 1 path per agent
 - Redundant path calculations
 - Waypoint-fighting
- local collision avoidance
 - RVO/movement planning
 - expensive at scale
 - Cellular automata
 - lacks fluidity of motion

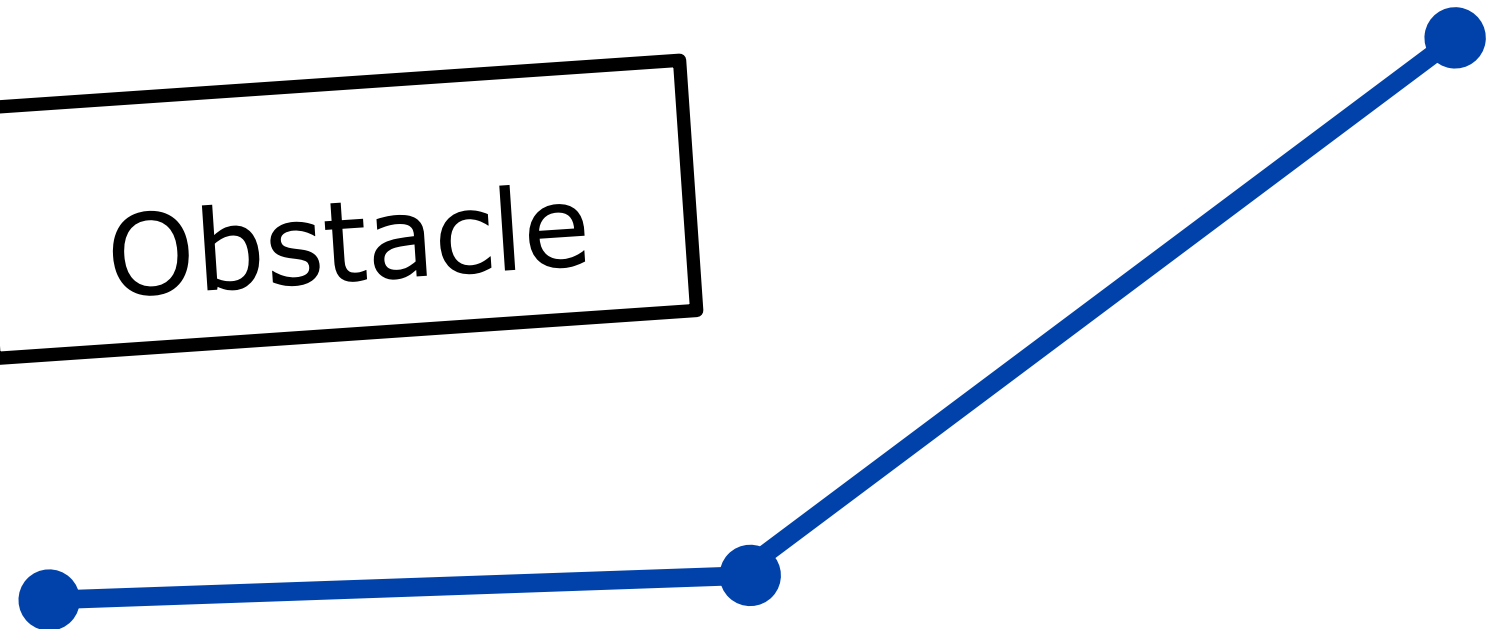
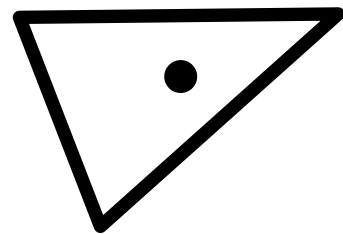
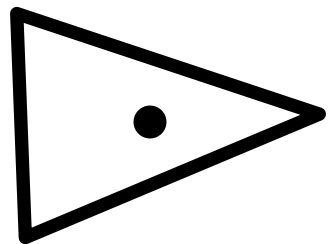
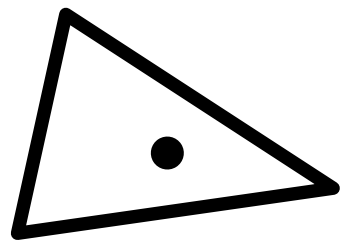
Path Following



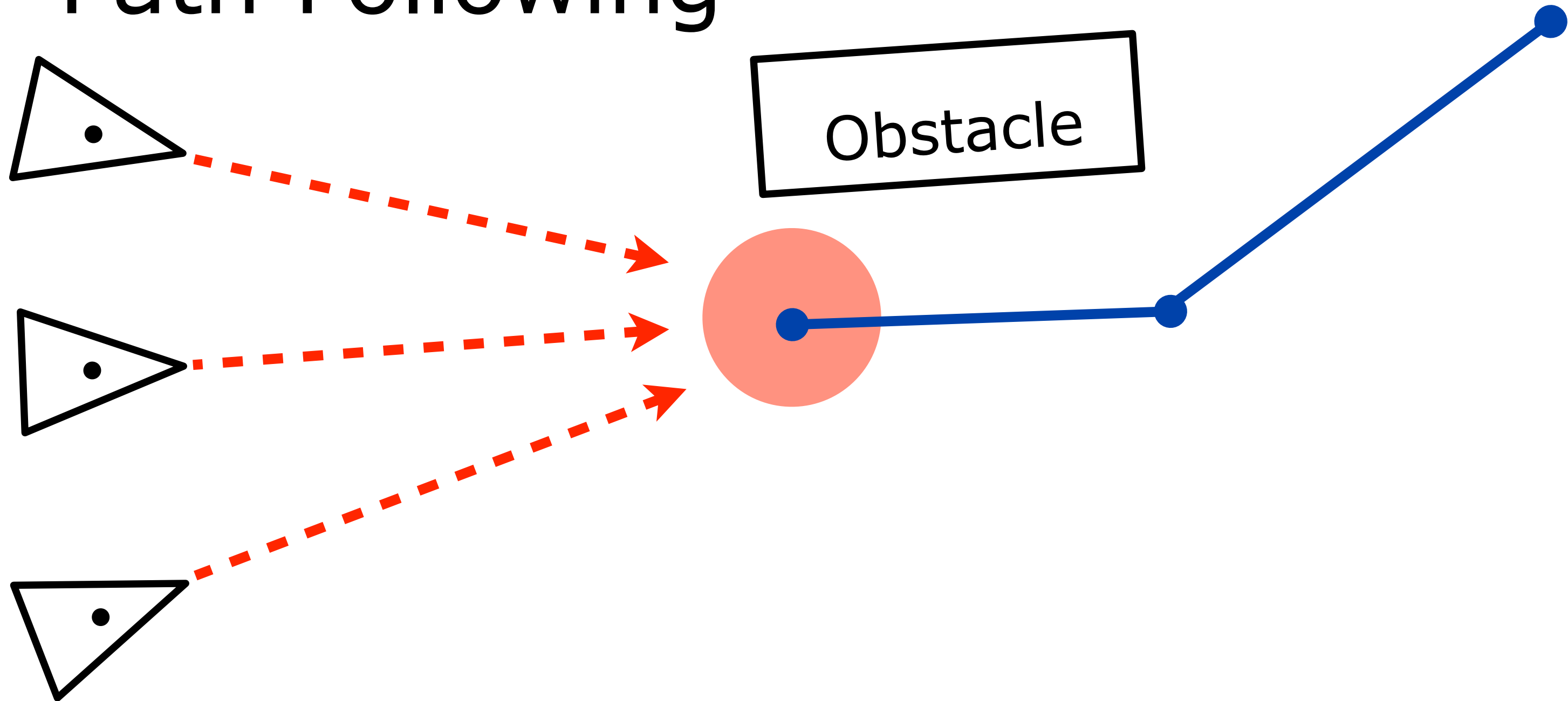
Path Following



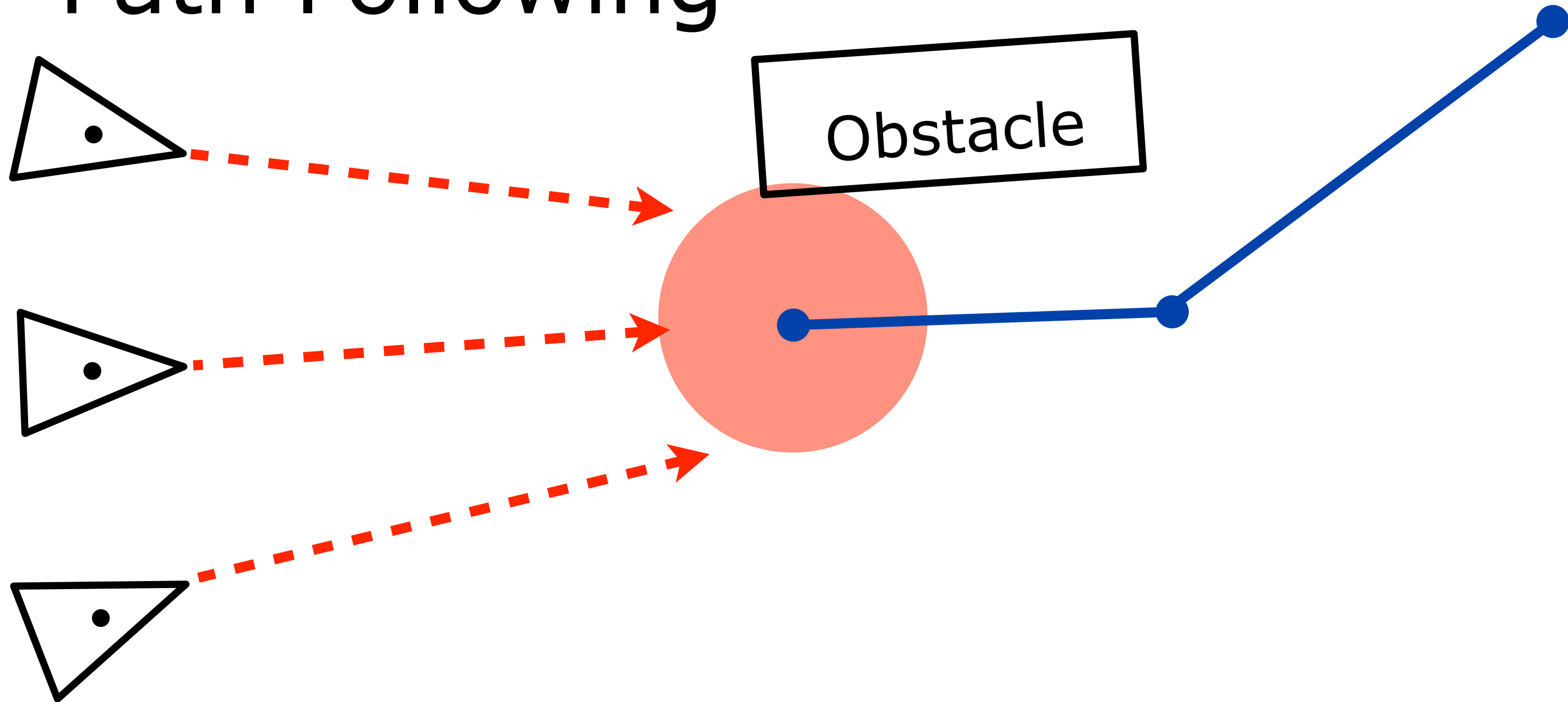
Path Following



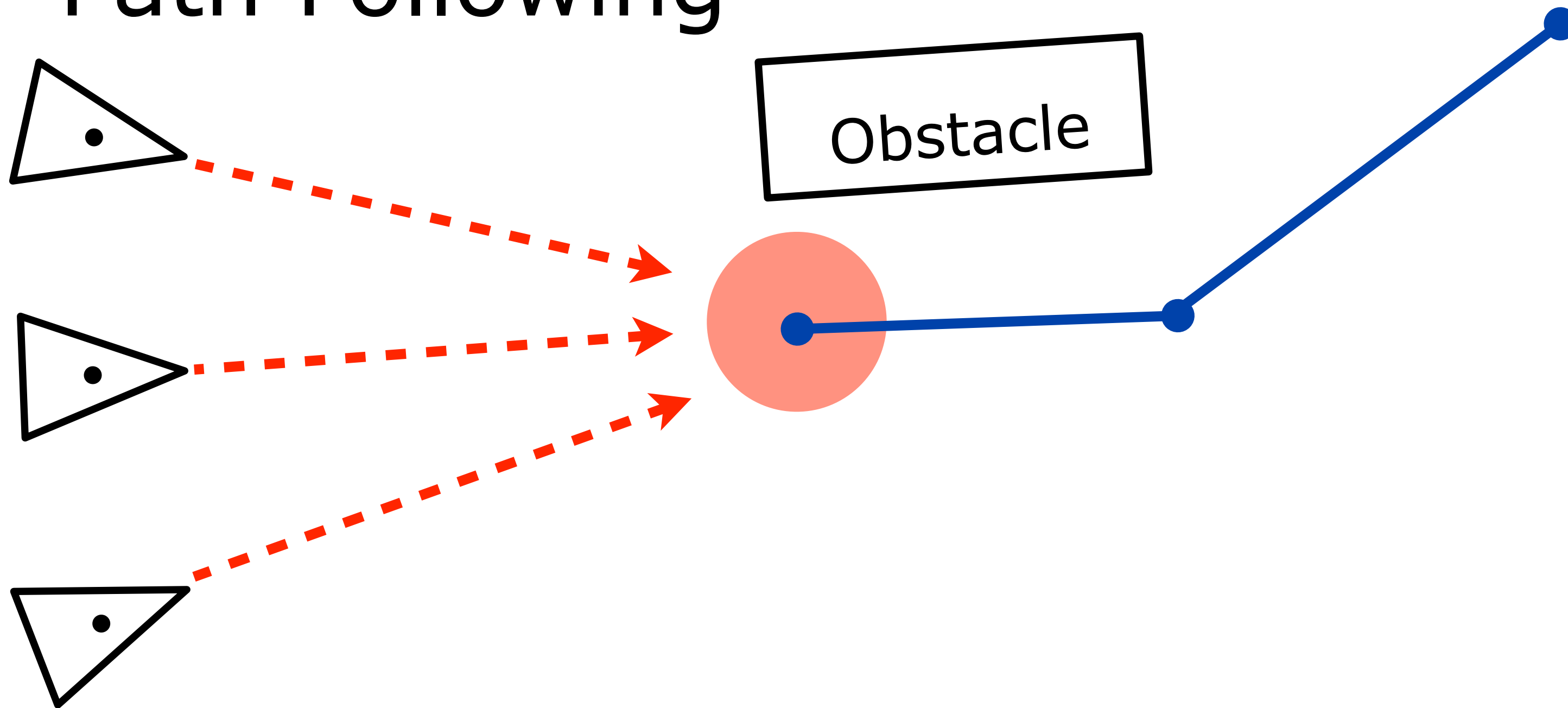
Path Following



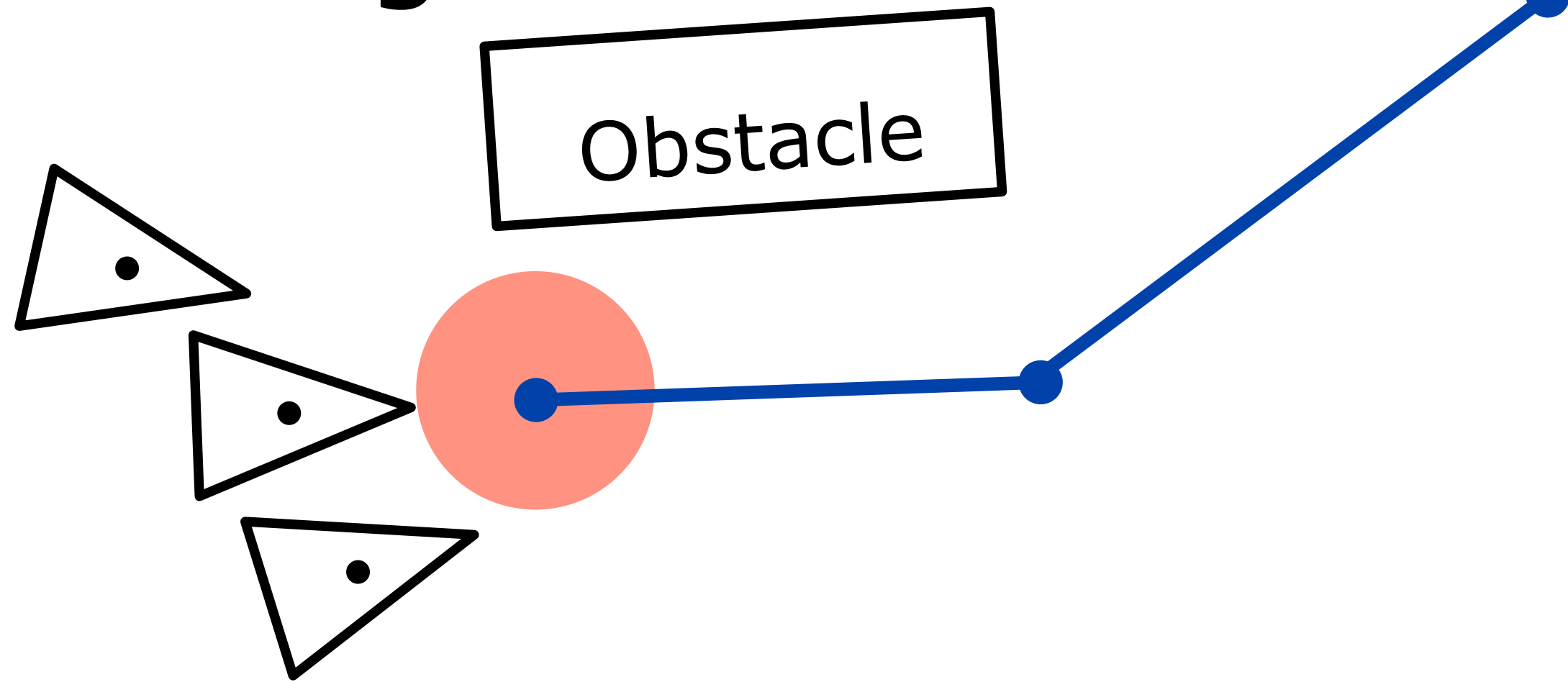
Path Following



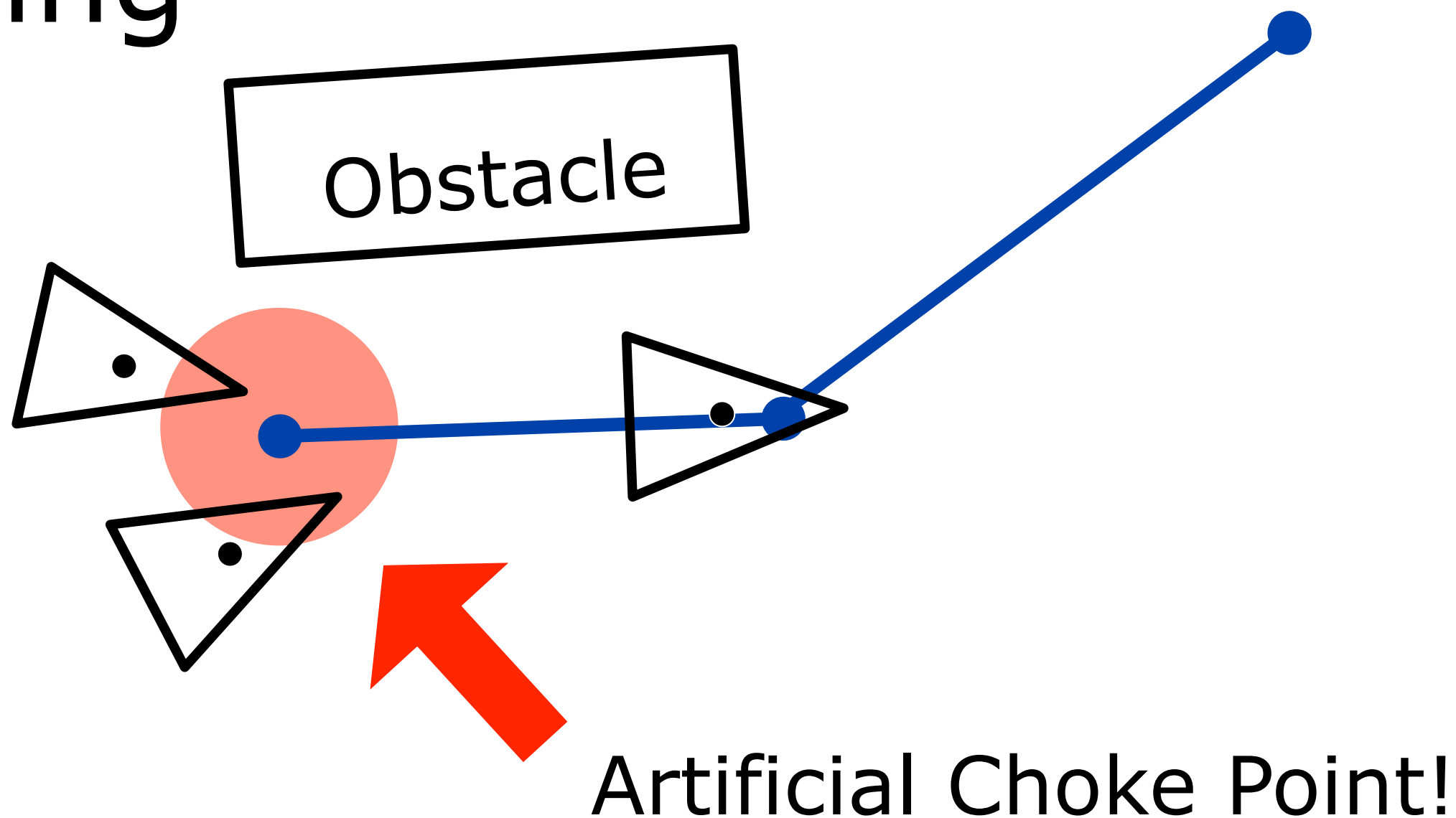
Path Following



Path Following

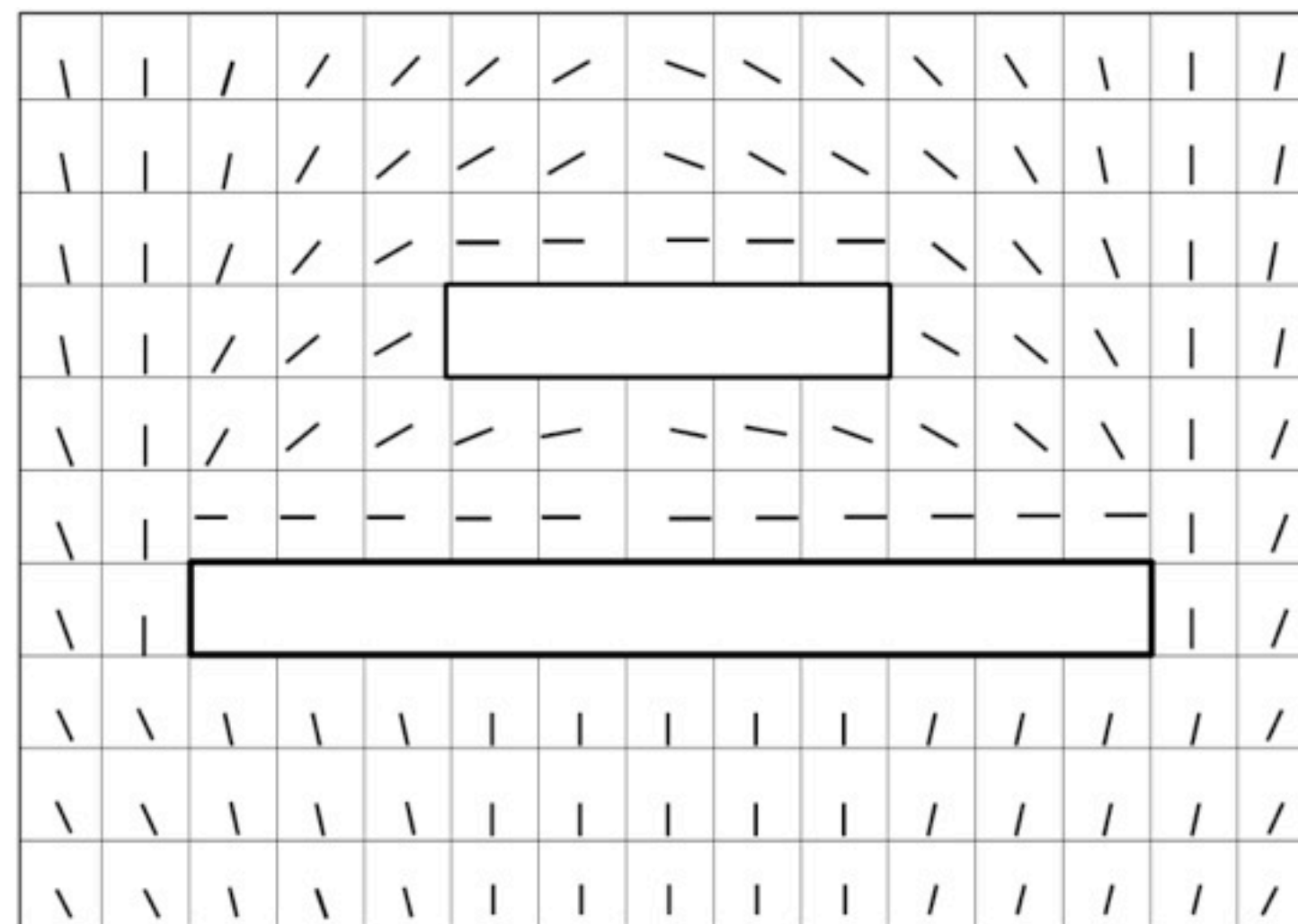


Path Following

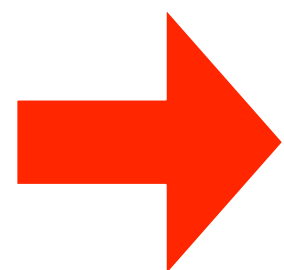


Flow Fields

- Discrete approximation of a 'flow function'
- Best path from every cell to closest goal
- Agents lookup path direction in flow field



Flow Field Generation



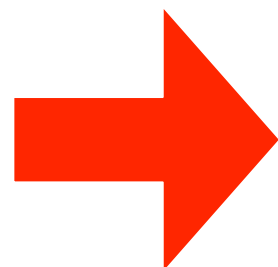
```
openList.addAll( grid.goalCells );
while ( !openList.isEmpty() ) {
    Cell c = openList.pop();

    for ( Cell n : c.neighbors ) {
        float alt = c.dist + distance( n, c );
        if ( alt < n.dist ) {
            n.dist = alt;
            n.flow = norm( c.pos - n.pos );
        }
    }
}
```


Flow Field Generation

```
openList.addAll( grid.goalCells );
while ( !openList.isEmpty() ) {
    Cell c = openList.pop();

    for ( Cell n : c.neighbors ) {
        float alt = c.dist + distance( n, c );
        if ( alt < n.dist ) {
            n.dist = alt;
            n.flow = norm( c.pos - n.pos );
        }
    }
}
```



Agents

Agents

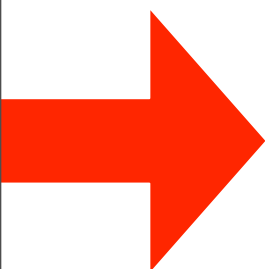
- Point mass
- Collision Circle
- Max Force
- Max Speed
- Neighbor Radius

Agents

- Point mass
- Collision Circle
- Max Force
- Max Speed
- Neighbor Radius

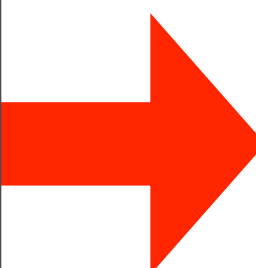
- Behaviors
 - Flow field following
 - Separation
 - Alignment
 - Cohesion

Flow field following



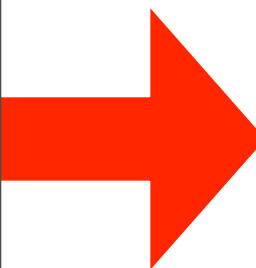
```
vec2 FlowFieldFollow(Agent agent, Grid grid) {  
    vec2 desired = grid.flowAtPoint(agent.position);  
  
    desired = desired * agent.maxSpeed;  
    desired -= agent.velocity;  
  
    return desired * agent.maxForce / agent.maxSpeed;  
}
```

Flow field following



```
vec2 FlowFieldFollow(Agent agent, Grid grid) {  
    vec2 desired = grid.flowAtPoint(agent.position);  
  
    desired = desired * agent.maxSpeed;  
    desired -= agent.velocity;  
  
    return desired * agent.maxForce / agent.maxSpeed;  
}
```

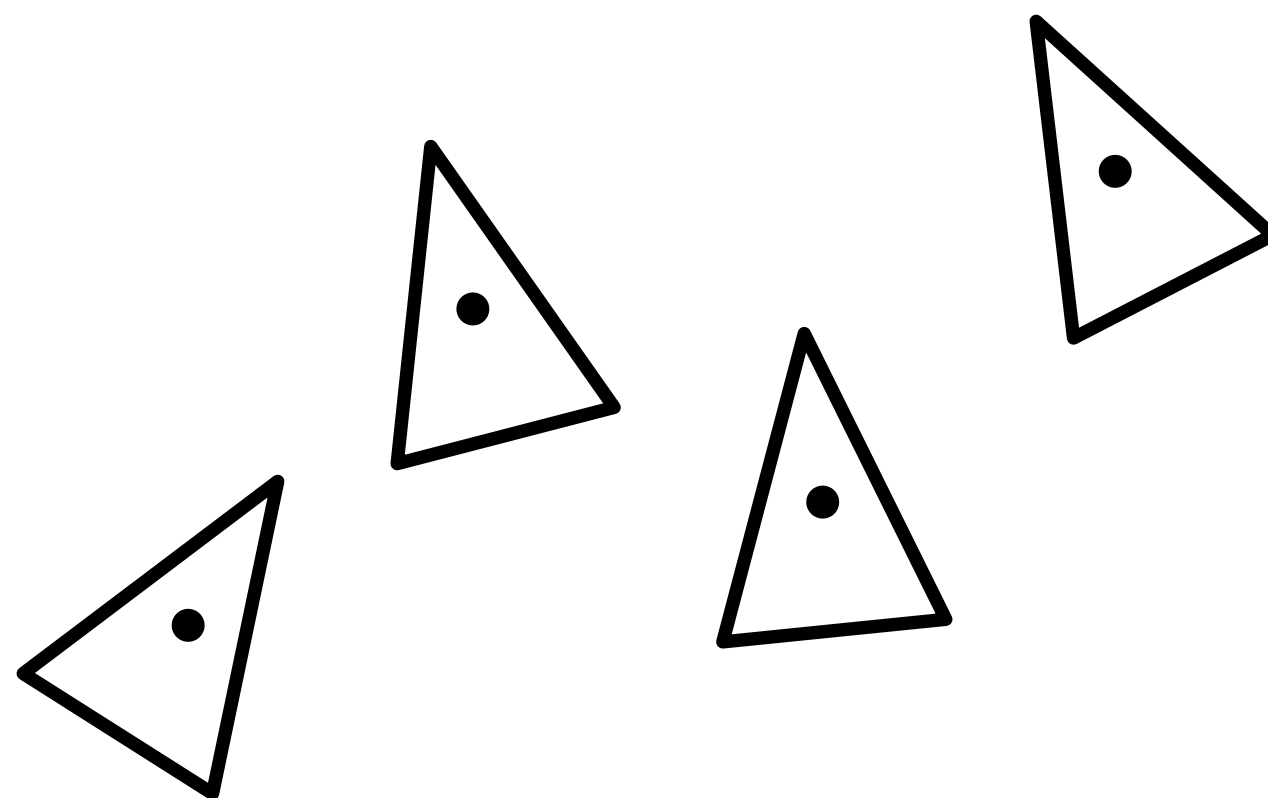
Flow field following



```
vec2 FlowFieldFollow(Agent agent, Grid grid) {  
    vec2 desired = grid.flowAtPoint(agent.position);  
  
    desired = desired * agent.maxSpeed;  
    desired -= agent.velocity;  
  
    return desired * agent.maxForce / agent.maxSpeed;  
}
```

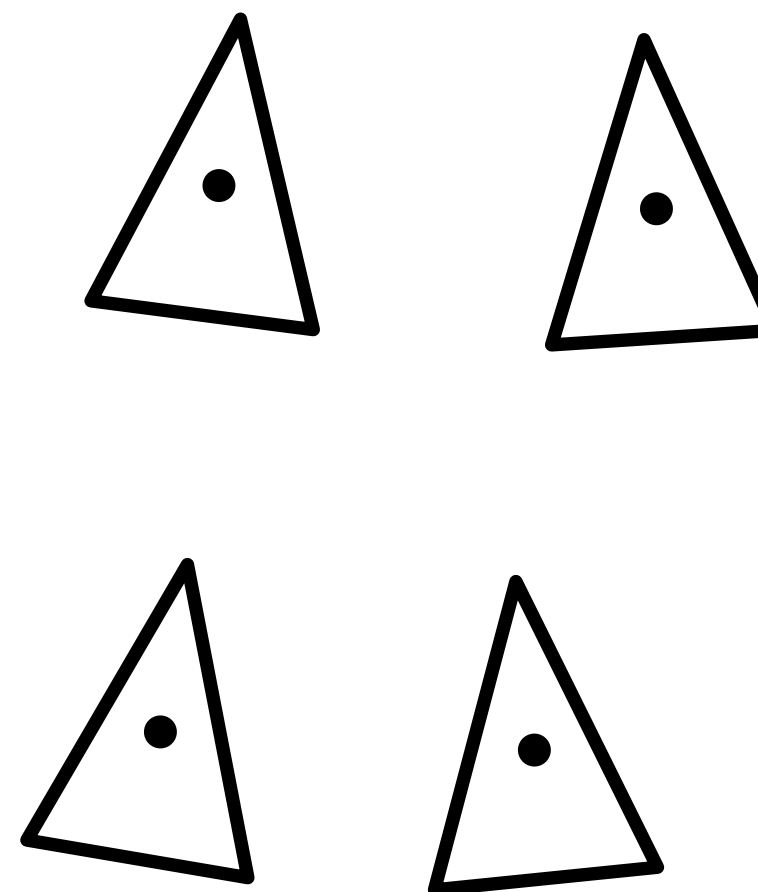

Flocking

- Move as a group
- Separation + Cohesion
 - Clustering
- Alignment
 - Face common direction

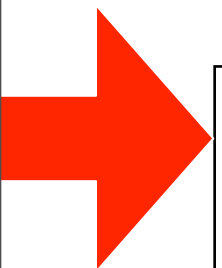


Flocking

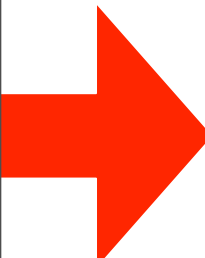
- Move as a group
- Separation + Cohesion
 - Clustering
- Alignment
 - Face common direction



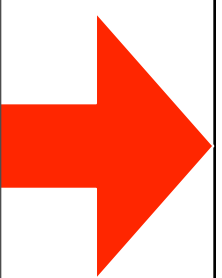
Separation



```
vec2 Separation(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 totalForce = vec2.zero;  
    for (Agent neighbor : neighbors) {  
        vec2 pushForce = (agent.pos - neighbor.pos)  
        totalForce += 1- (pushForce / agent.neighborRadius);  
    }  
  
    totalForce /= neighbors.count();  
    totalForce *= agent.maxForce;  
}
```

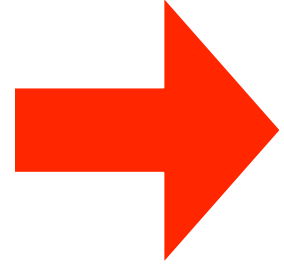


```
vec2 Separation(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 totalForce = vec2.zero;  
    for (Agent neighbor : neighbors) {  
        vec2 pushForce = (agent.pos - neighbor.pos)  
        totalForce += 1- (pushForce / agent.neighborRadius);  
    }  
  
    totalForce /= neighbors.count();  
    totalForce *= agent.maxForce;  
}
```

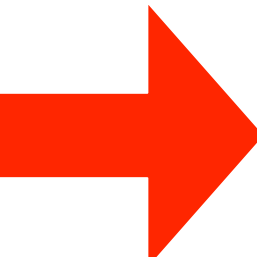


```
vec2 Separation(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 totalForce = vec2.zero;  
    for (Agent neighbor : neighbors) {  
        vec2 pushForce = (agent.pos - neighbor.pos)  
        totalForce += 1- (pushForce / agent.neighborRadius);  
    }  
  
    totalForce /= neighbors.count();  
    totalForce *= agent.maxForce;  
}
```

Cohesion

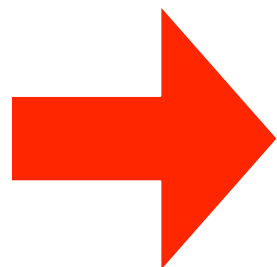


```
vec2 Cohesion(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 centerOfMass = agent.position;  
    for (Agent neighbor : neighbors)  
        centerOfMass += neighbor.position;  
    centerOfMass /= neighbors.count();  
  
    vec2 desired = centerOfMass - agent.position;  
    desired *= agent.maxSpeed / desired.mag();  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```

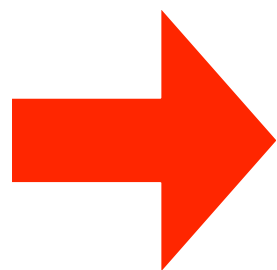



```
vec2 Cohesion(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 centerOfMass = agent.position;  
    for (Agent neighbor : neighbors)  
        centerOfMass += neighbor.position;  
    centerOfMass /= neighbors.count();  
  
    vec2 desired = centerOfMass - agent.position;  
    desired *= agent.maxSpeed / desired.mag();  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```

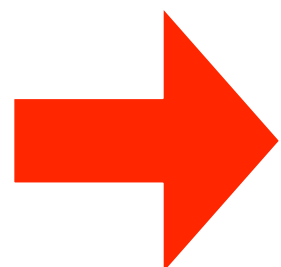
```
vec2 Cohesion(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 centerOfMass = agent.position;  
    for (Agent neighbor : neighbors)  
        centerOfMass += neighbor.position;  
    centerOfMass /= neighbors.count();  
  
    vec2 desired = centerOfMass - agent.position;  
    desired *= agent.maxSpeed / desired.mag();  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```



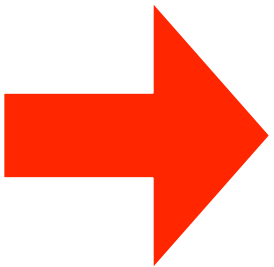
```
vec2 Cohesion(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 centerOfMass = agent.position;  
    for (Agent neighbor : neighbors)  
        centerOfMass += neighbor.position;  
    centerOfMass /= neighbors.count();  
  
    vec2 desired = centerOfMass - agent.position;  
    desired *= agent.maxSpeed / desired.mag();  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```



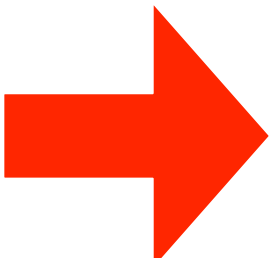
Alignment



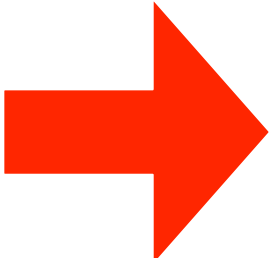
```
vec2 Alignment(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 avgHeading = norm( agent.velocity );  
    for (Agent neighbor : neighbors)  
        avgHeading += norm( neighbor.velocity );  
    avgHeading /= neighbors.count();  
  
    vec2 desired = avgHeading * agent.maxSpeed;  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```



```
vec2 Alignment(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 avgHeading = norm( agent.velocity );  
    for (Agent neighbor : neighbors)  
        avgHeading += norm( neighbor.velocity );  
    avgHeading /= neighbors.count();  
  
    vec2 desired = avgHeading * agent.maxSpeed;  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```



```
vec2 Alignment(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 avgHeading = norm( agent.velocity );  
    for (Agent neighbor : neighbors)  
        avgHeading += norm( neighbor.velocity );  
    avgHeading /= neighbors.count();  
  
    vec2 desired = avgHeading * agent.maxSpeed;  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```



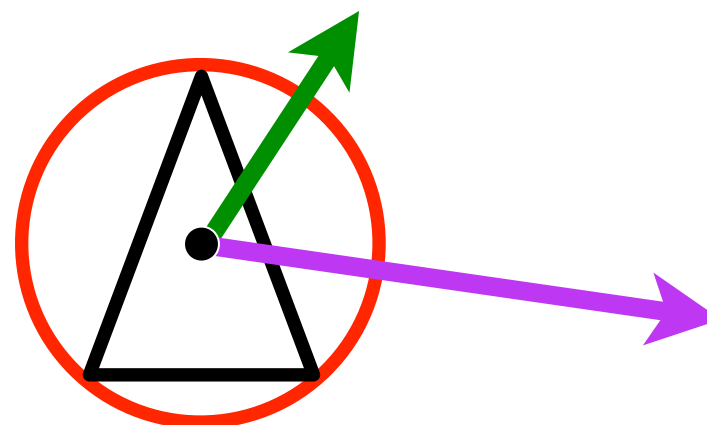
```
vec2 Alignment(Agent agent, List<Agent> neighbors) {  
    if (neighbors.empty()) return vec2.zero;  
  
    vec2 avgHeading = norm( agent.velocity );  
    for (Agent neighbor : neighbors)  
        avgHeading += norm( neighbor.velocity );  
    avgHeading /= neighbors.count();  
  
    vec2 desired = avgHeading * agent.maxSpeed;  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```


Benefits

- Pure functions
 - Defined in terms of arguments
 - No state or side effects
 - Concurrency is trivial & lock-free
- Flexible
 - Complexity through composition
 - Dynamically modify (LOD)
- Efficient
 - SIMD & Prioritization

Debugging

- Visual debugging information mandatory!
 - Collision shape
 - Movement direction
 - Net steering force
 - Current path direction

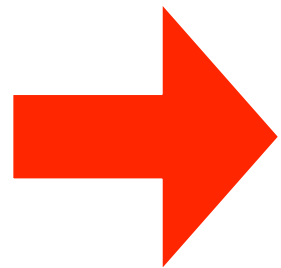


Current Stigma

- Inherently unstable
- Necessitates constant tweaking and bugfixing
- Stems from poor implementations

Seek

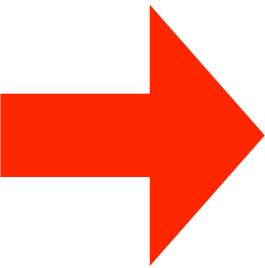
- Universally incorrect, except “Artificial Intelligence for Games” by Millington & Funge



```
vec2 seek(vec2 target, Agent a) {  
    vec2 desired = target - a.position;  
    desired *= a.maxSpeed / desired.mag();  
  
    vec2 force = desired - a.velocity;  
    return force; // WRONG - This is a velocity!  
}
```

Seek

- Universally incorrect, except “Artificial Intelligence for Games” by Millington & Funge



```
vec2 seek(vec2 target, Agent a) {  
    vec2 desired = target - a.position;  
    desired *= a.maxSpeed / desired.mag();  
  
    vec2 force = desired - a.velocity;  
    return force; // WRONG - This is a velocity!  
}
```

Seek

- Correct implementation defined in terms of max force

```
vec2 seek(vec2 target, Agent agent) {  
    vec2 desired = target - agent.position;  
    desired *= agent.maxSpeed / desired.mag();  
  
    vec2 force = desired - agent.velocity;  
    return force * (agent.maxForce / agent.maxSpeed);  
}
```

Flow fields + Steering

- Inexpensive, robust crowd simulation
- Flowfields
 - Best suited for common destinations in reasonably static environments
 - minimizes pathfinding calculations
- Steering
 - Behaviors Model specific kinds of movement
 - Complexity through composition

Graham Pentheny

graham.pentheny@gmail.com
@grahamboree

slides available at:
grahampentheny.com/gdc

Steering with Context Behaviours

Andrew Fray
Programmer, Spry Fox

Andrew Fray



SPRYFOX



Steering behaviours



Steering behaviours

- Lightweight framework



Steering behaviours

- Lightweight framework
- Simple behaviours



Steering behaviours

- Lightweight framework
- Simple behaviours
- Emergent behaviour



Drawbacks

Drawbacks

- No guaranteed movement constraint

Drawbacks

- No guaranteed movement constraint
- Inconsistent decisions

Drawbacks

- No guaranteed movement constraint
- Inconsistent decisions



<http://www.flickr.com/photos/sanithomas/6063443802/>

Drawbacks

- No guaranteed movement constraint
- Inconsistent decisions



<http://www.flickr.com/photos/sanithomas/6063443802/>



<http://www.flickr.com/photos/psykotrooper/7529582638/>

Drawbacks

- No guaranteed movement constraint
- Inconsistent decisions



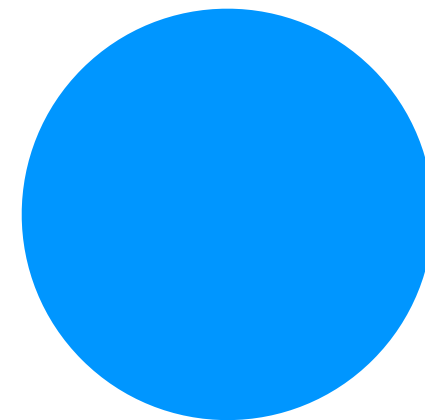
<http://www.flickr.com/photos/sanithomas/6063443802/>



<http://www.flickr.com/photos/97302051@N00/2521446258/>

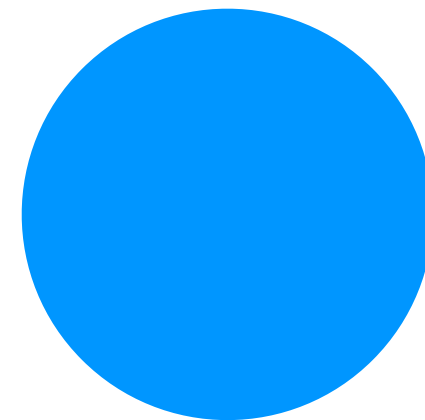
Balanced vectors problem

Balanced vectors problem



Balanced vectors problem

Chase behaviour



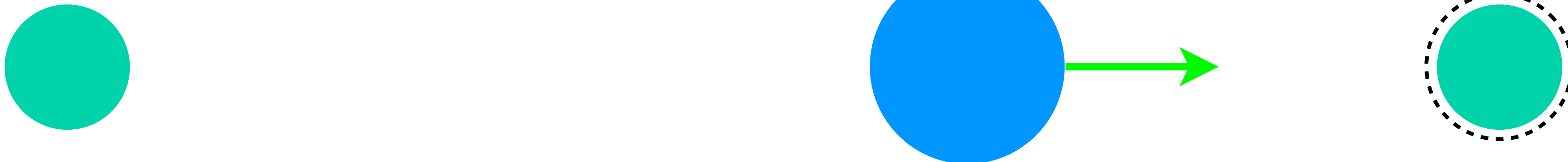
Balanced vectors problem

Chase behaviour



Balanced vectors problem

Chase behaviour



Balanced vectors problem

Chase behaviour

Avoid behaviour



Balanced vectors problem

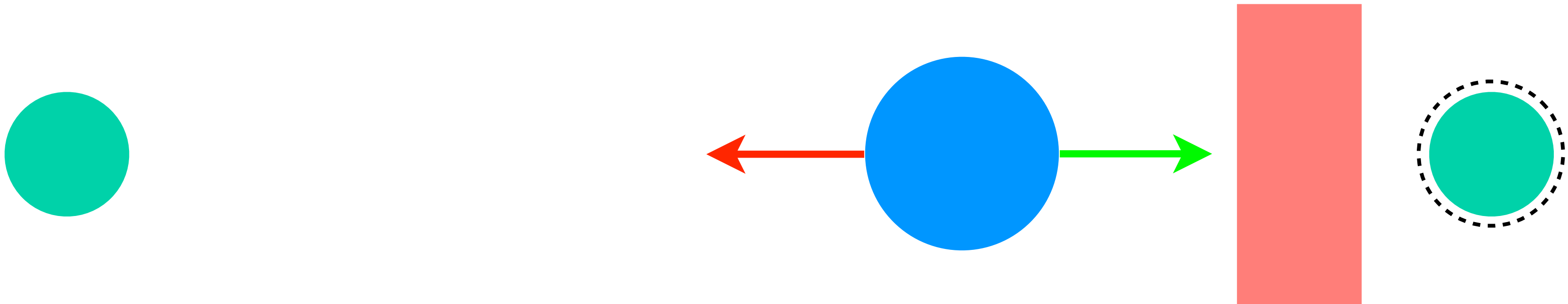
Chase behaviour

Avoid behaviour



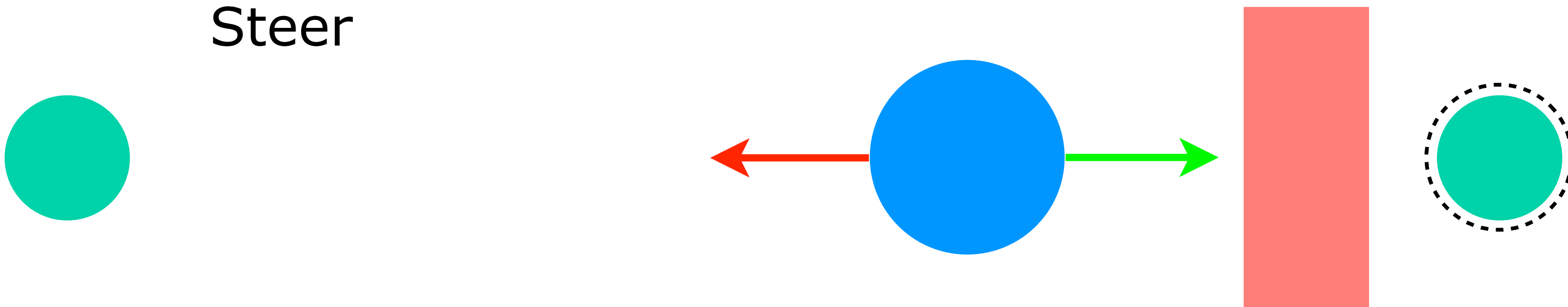
Balanced vectors problem

Chase behaviour
Avoid behaviour



Balanced vectors problem

Chase behaviour
Avoid behaviour
Steer

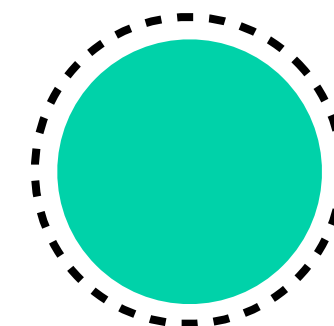
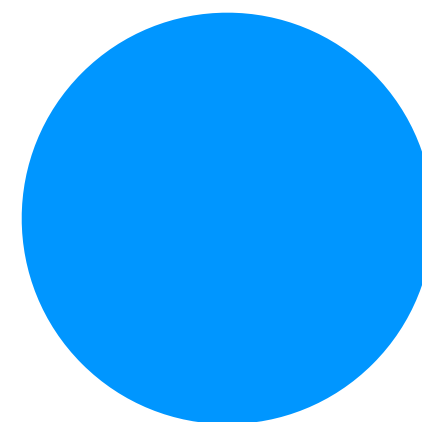
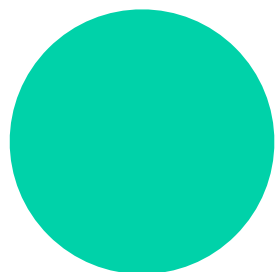


Balanced vectors problem

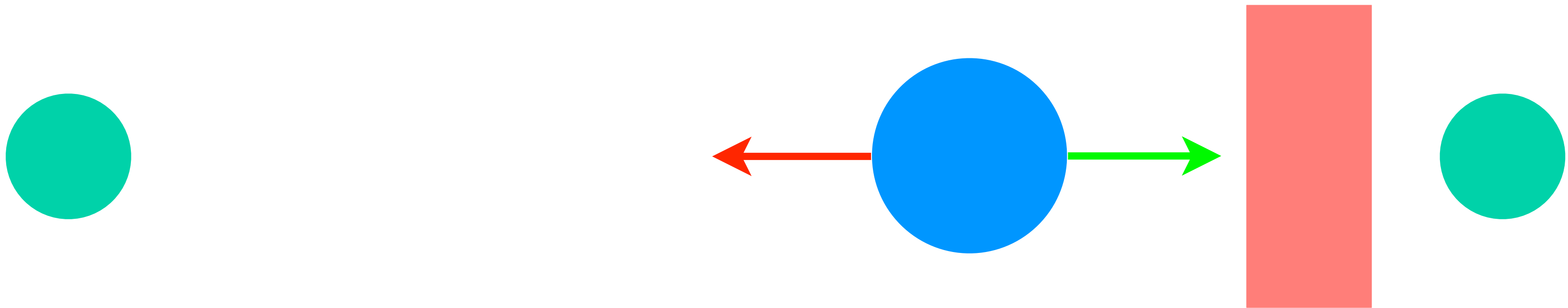
Chase behaviour

Avoid behaviour

Steer

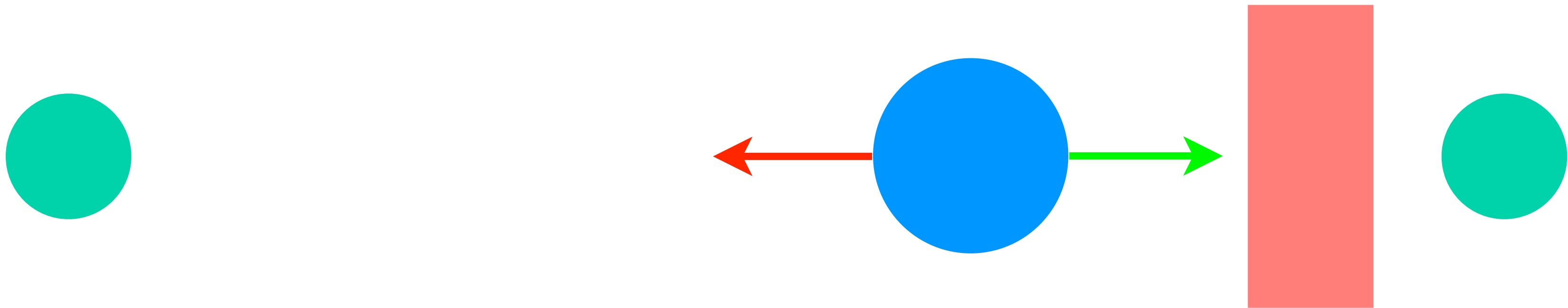


"Solutions"



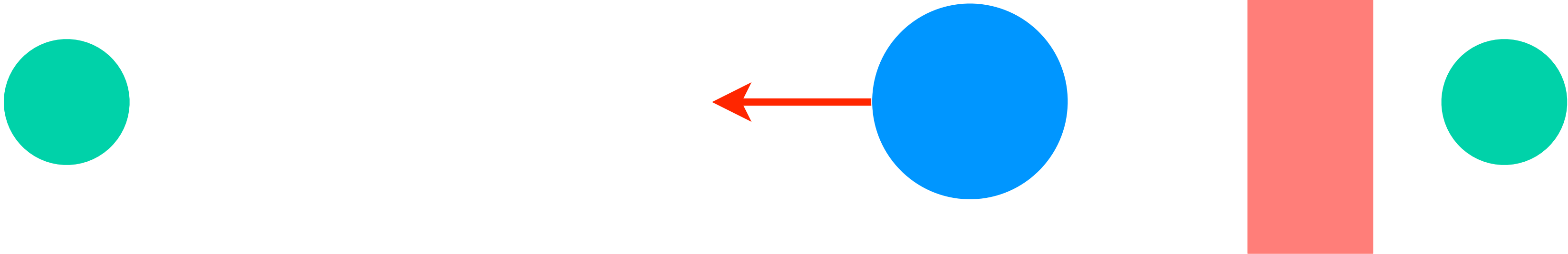
“Solutions”

- Ignore Chase



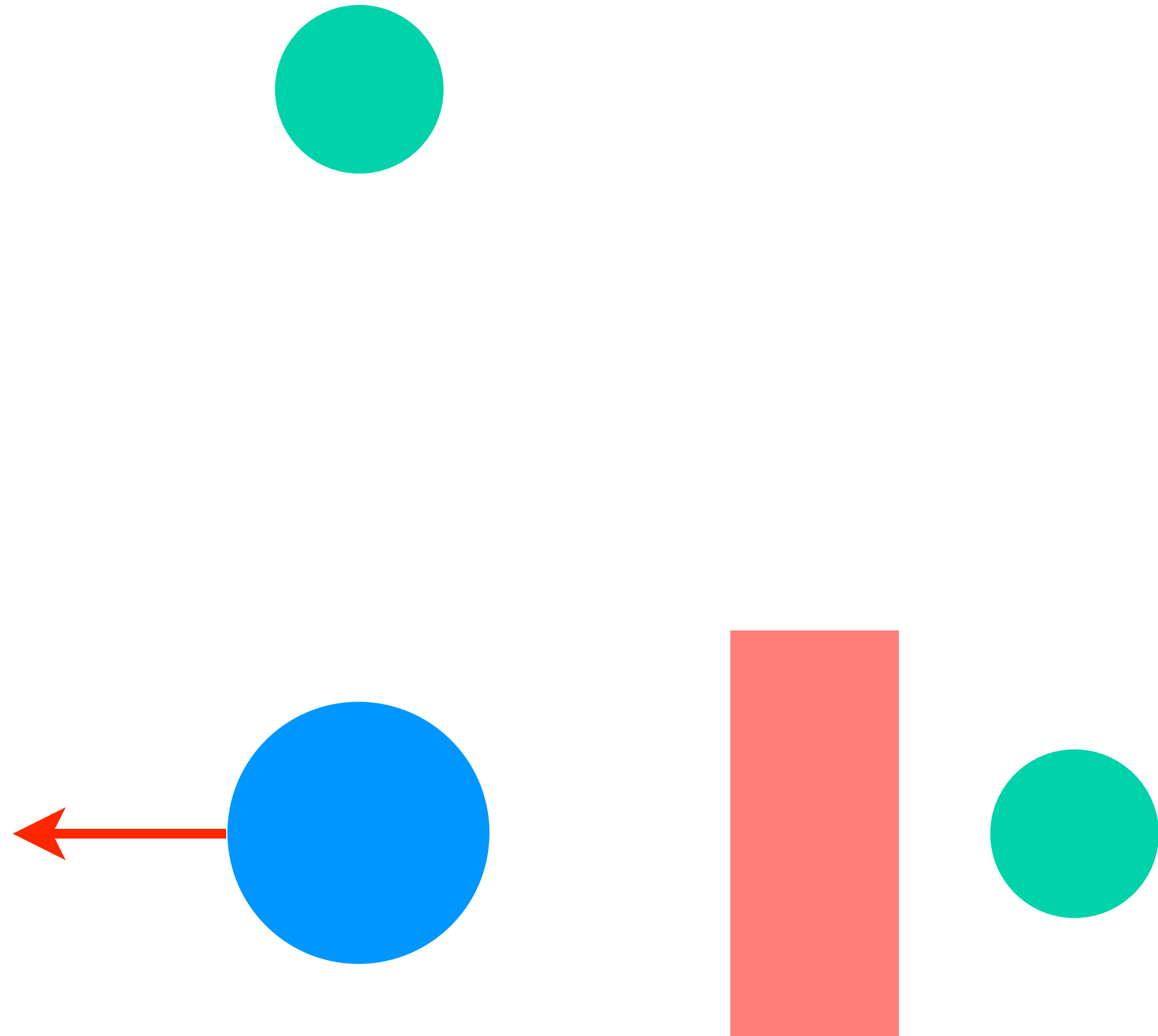
“Solutions”

- Ignore Chase



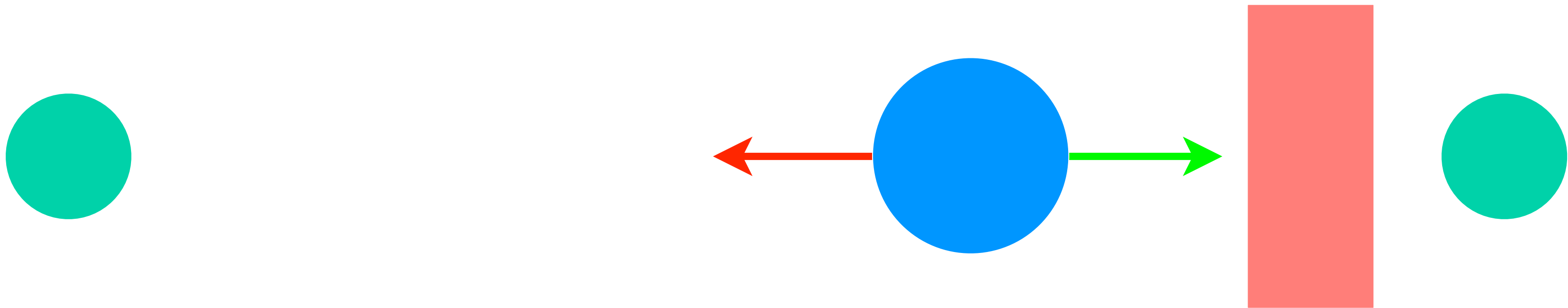
“Solutions”

- Ignore Chase



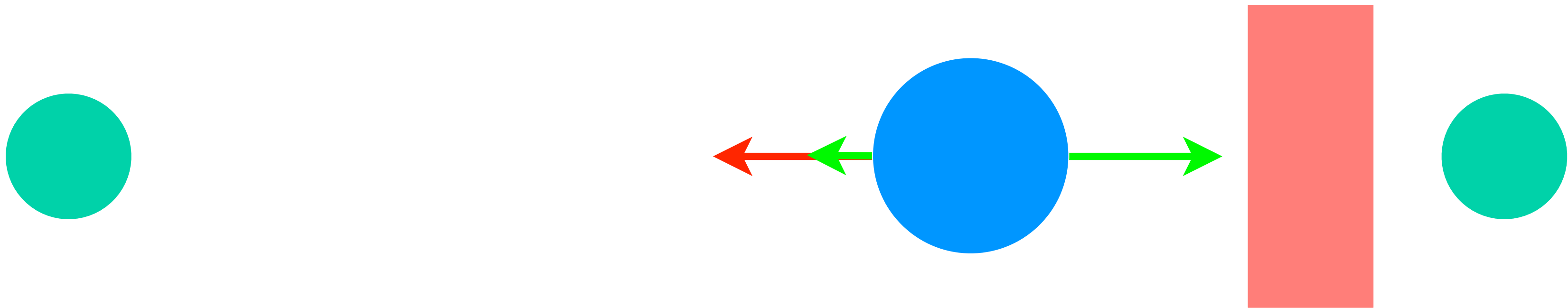
“Solutions”

- Ignore Chase
- Chase both targets



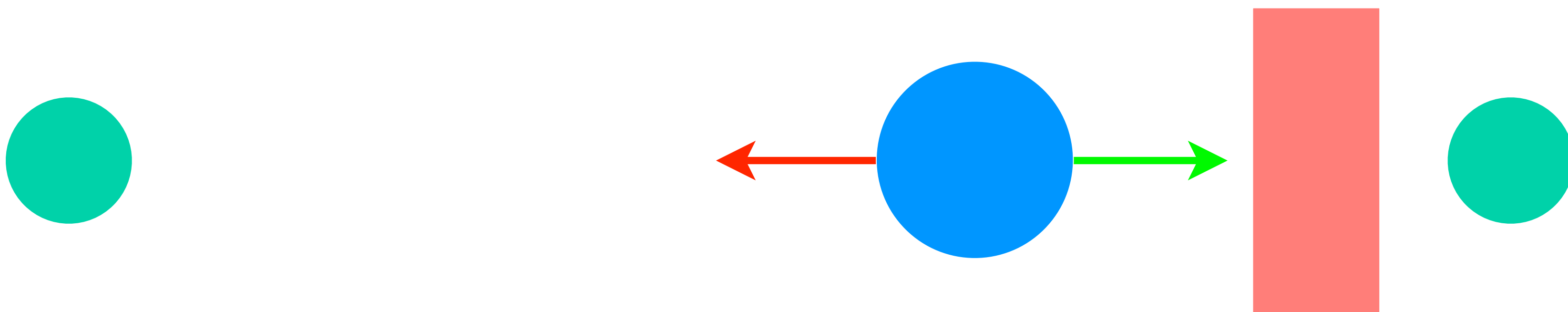
“Solutions”

- Ignore Chase
- Chase both targets



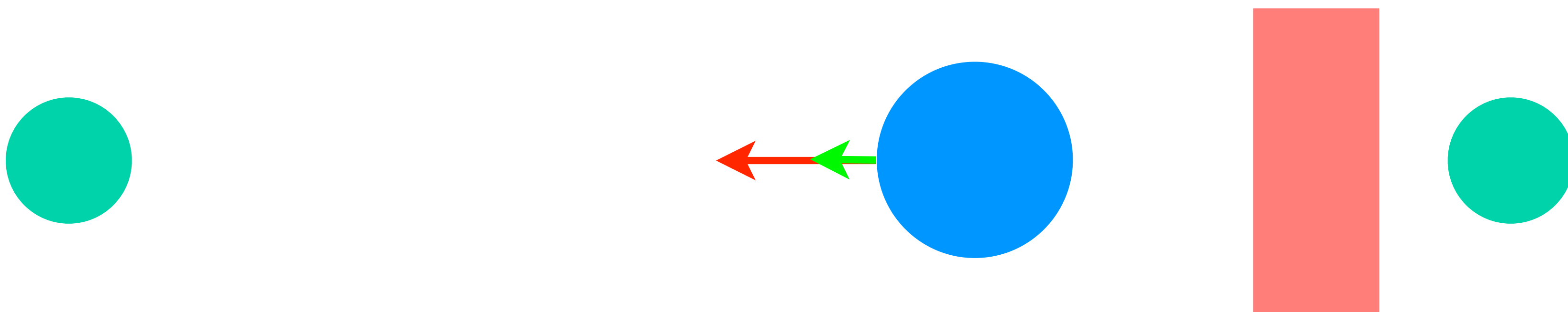
“Solutions”

- Ignore Chase
- Chase both targets
- Validate target before chase



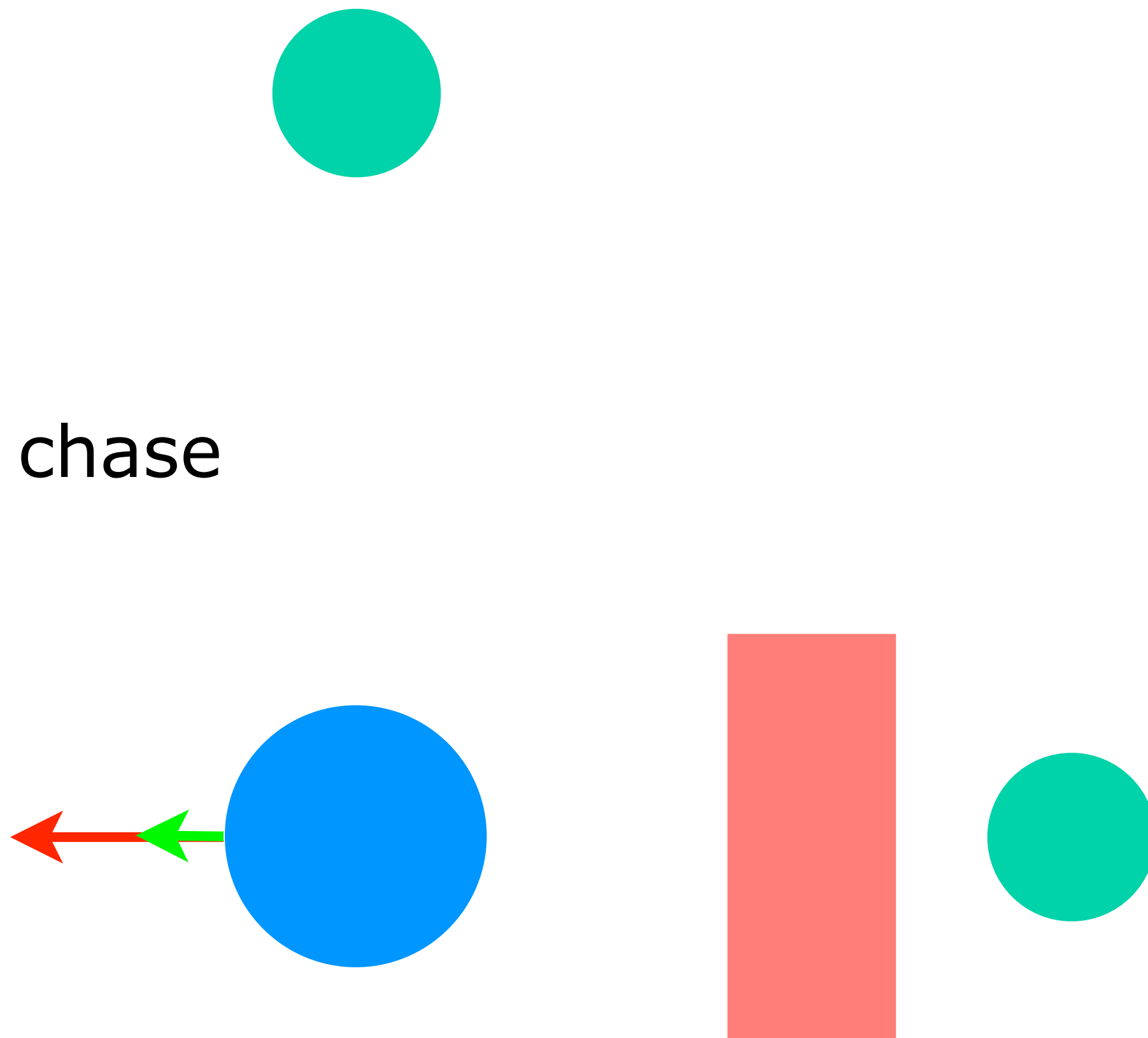
“Solutions”

- Ignore Chase
- Chase both targets
- Validate target before chase



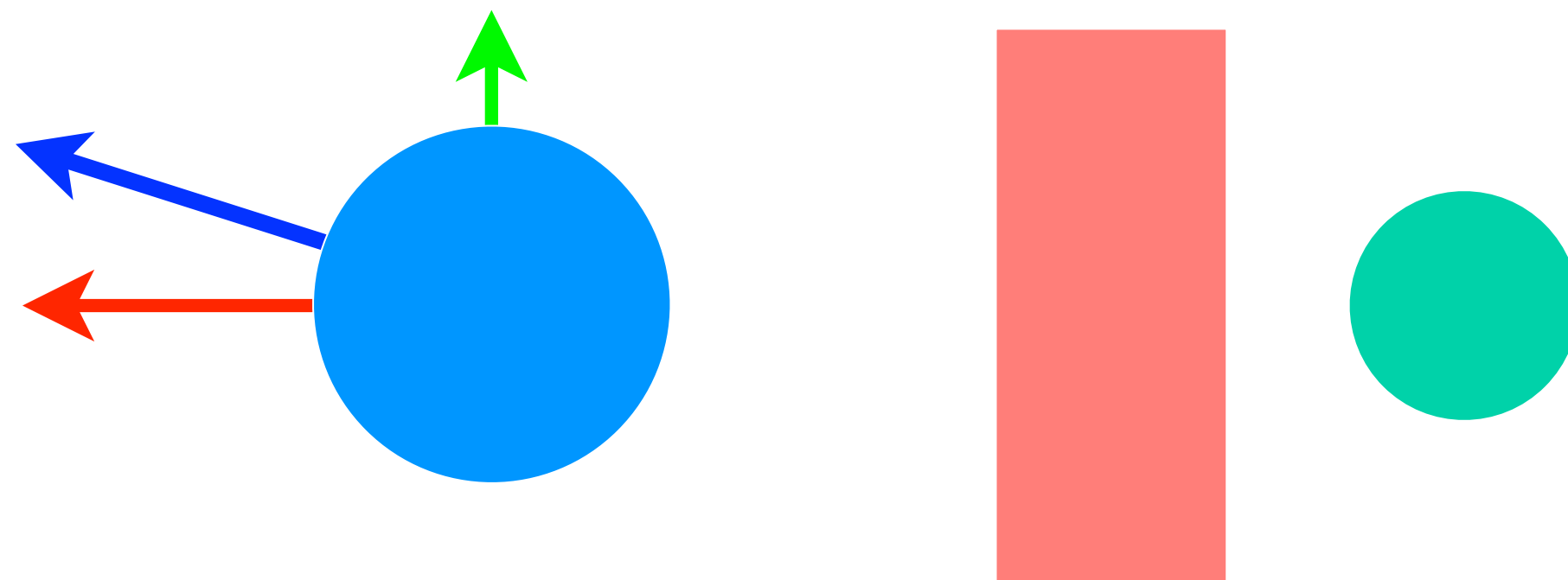
“Solutions”

- Ignore Chase
- Chase both targets
- Validate target before chase

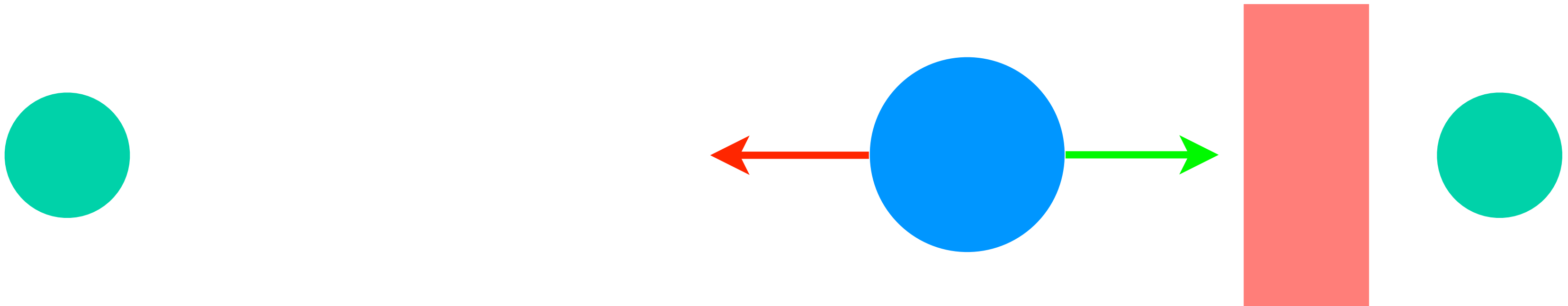


“Solutions”

- Ignore Chase
- Chase both targets
- Validate target before chase



Design Flaw



Merge contexts, not decisions

Context Behaviour System

Context Controller

Context Behaviour System

Context Controller

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour System

Context Controller

Context Map

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour System

Context Controller

Interest Map

Danger Map

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour System

Context Controller

Interest Map

Danger Map

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour

Context Behaviour System

Context Controller

RESULT

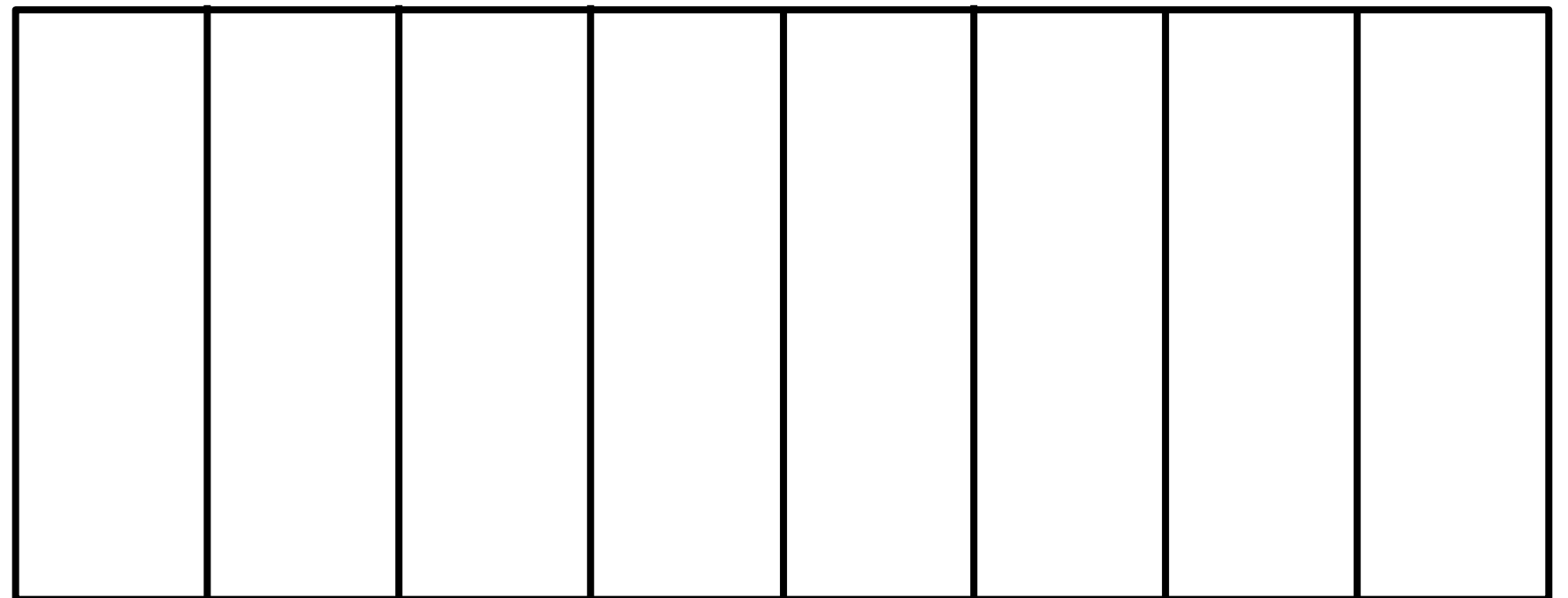
Context Behaviour

Context Behaviour

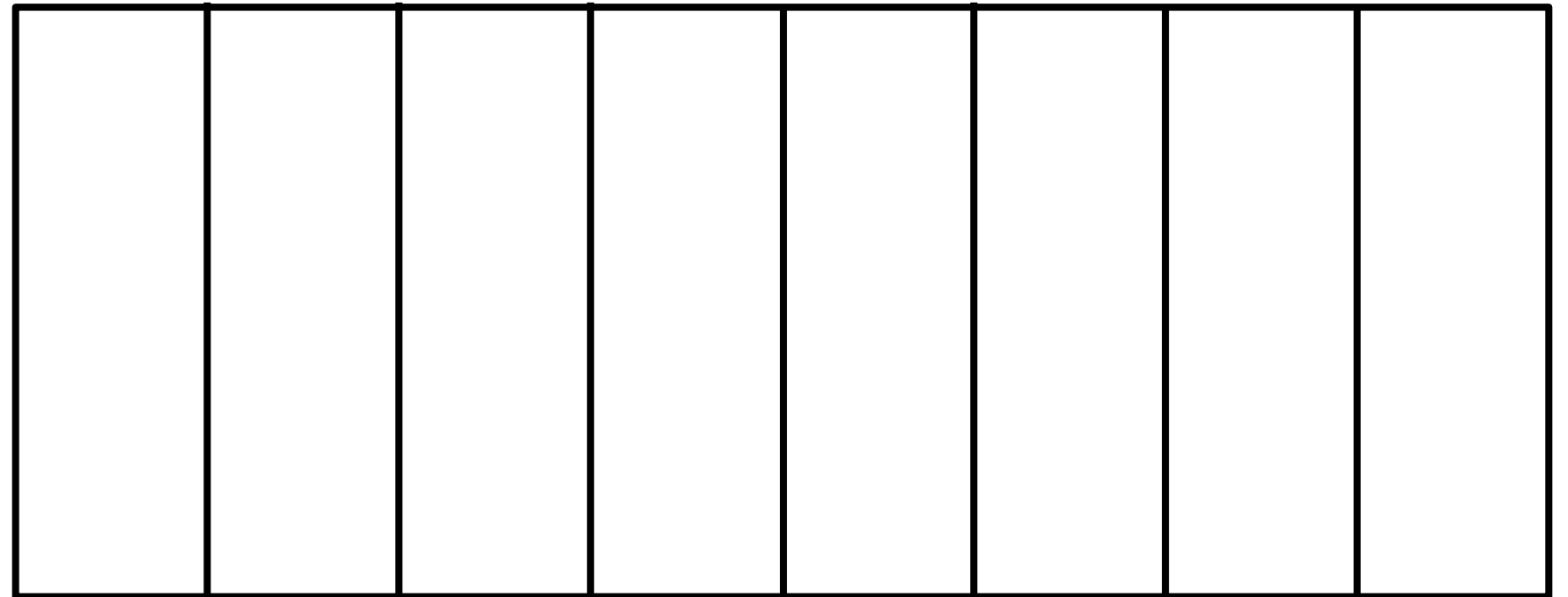
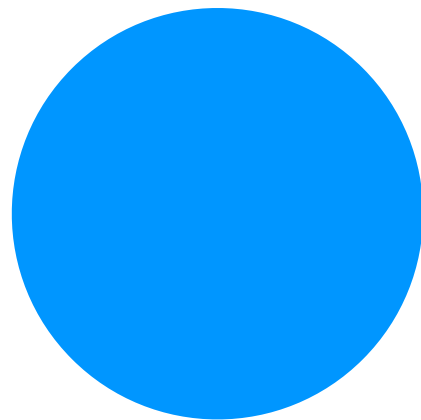
Context Behaviour

Context Behaviour

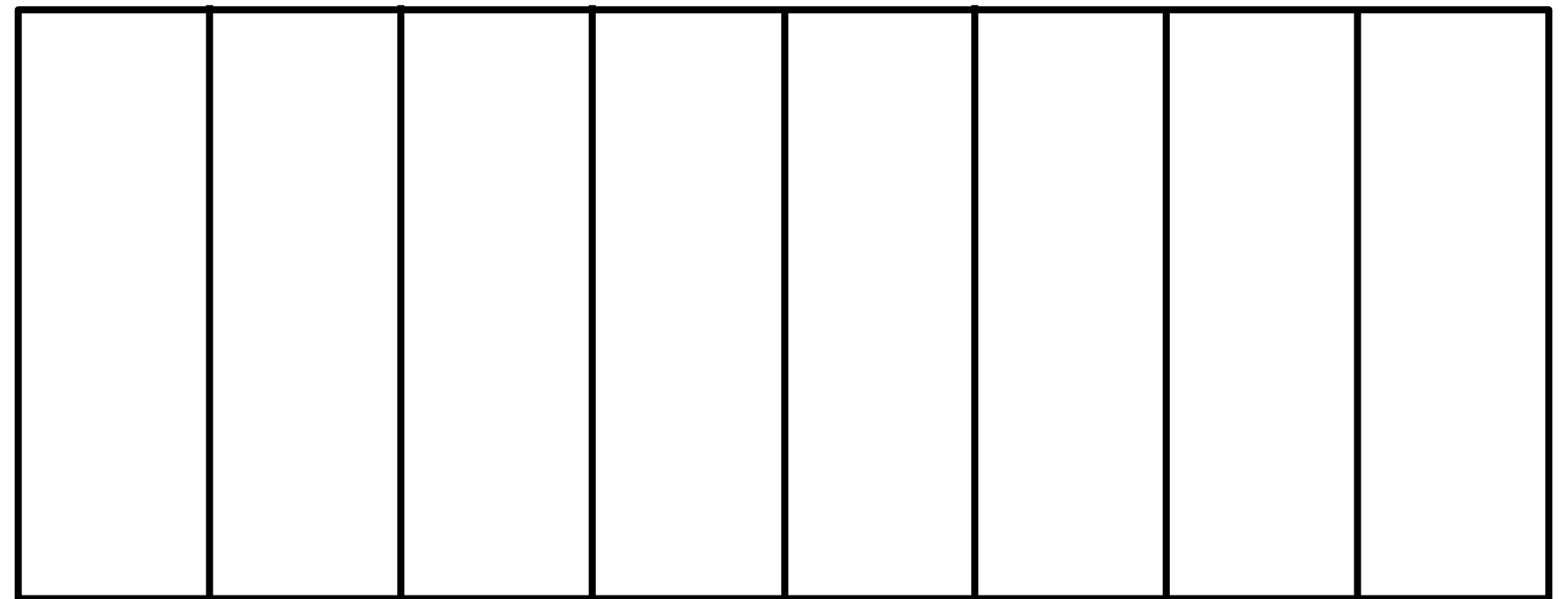
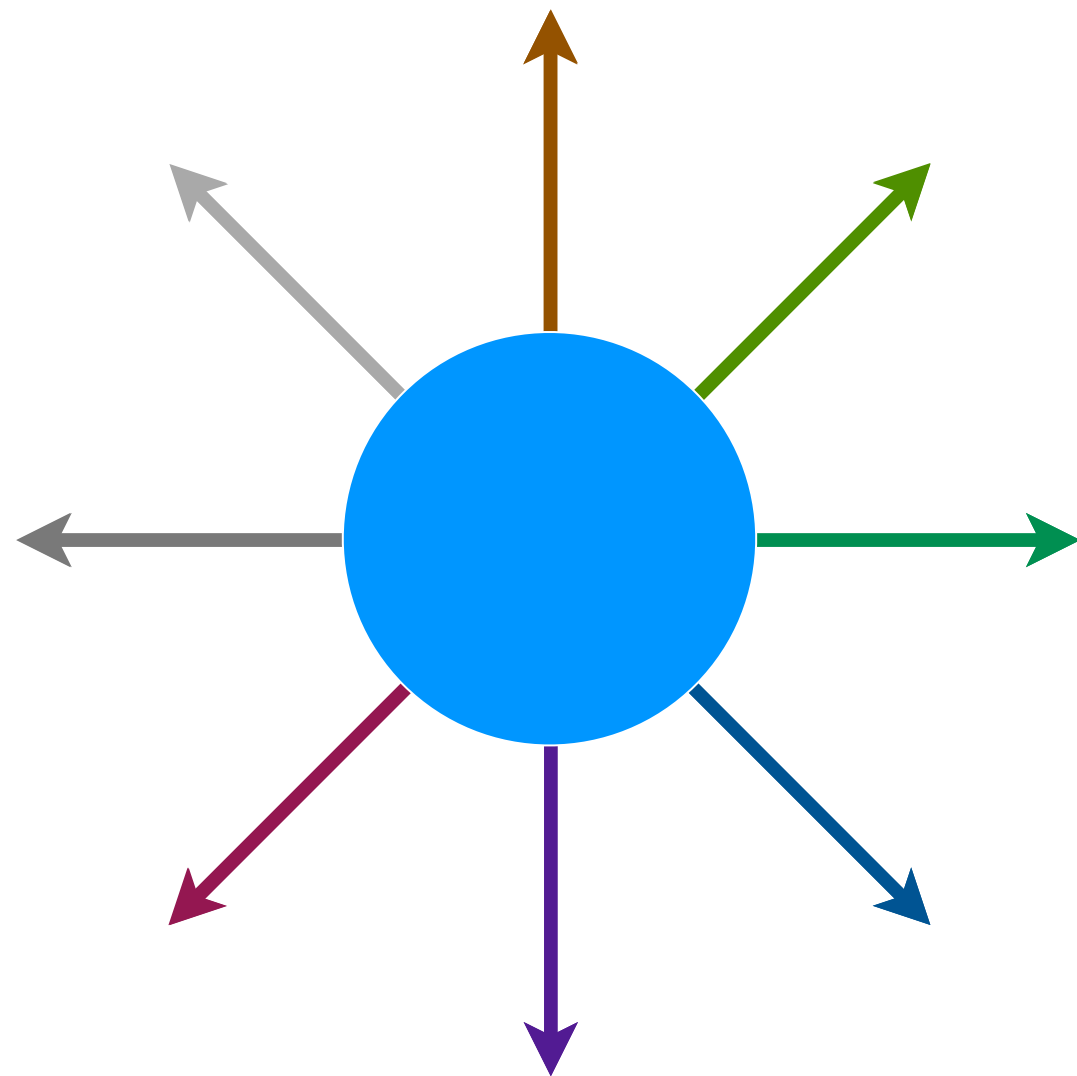
Context maps



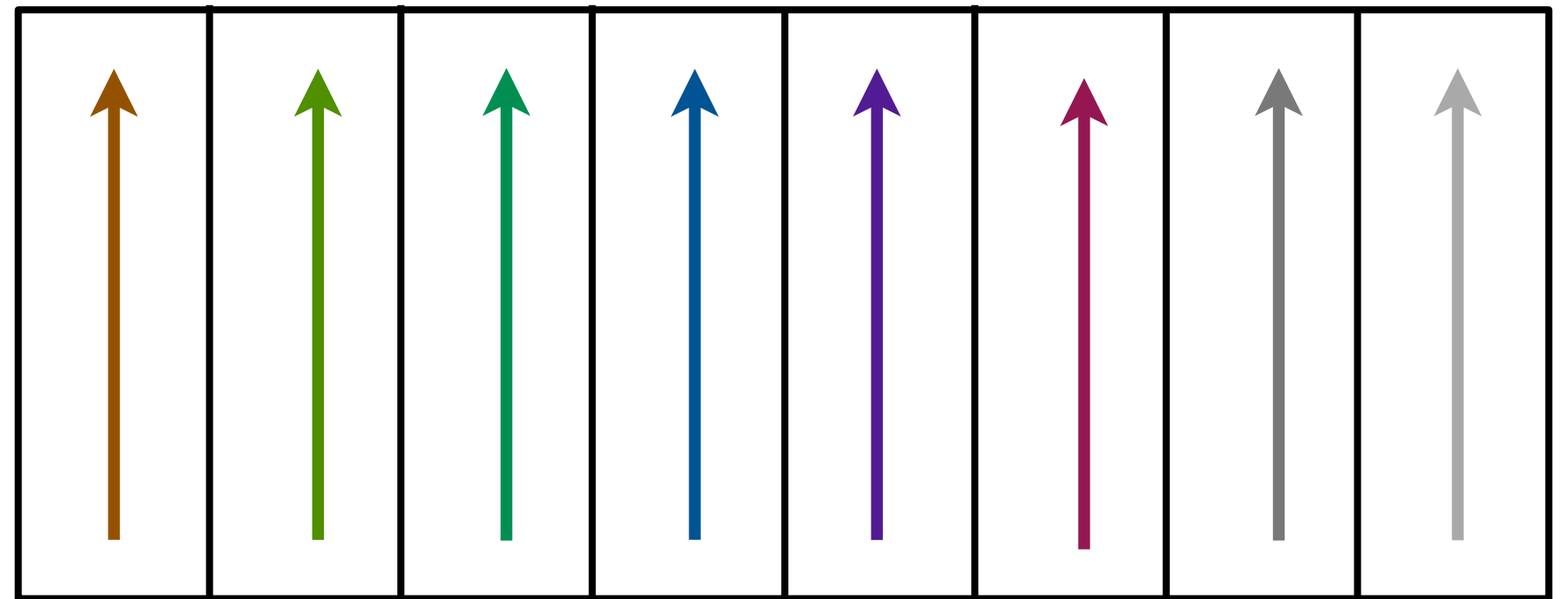
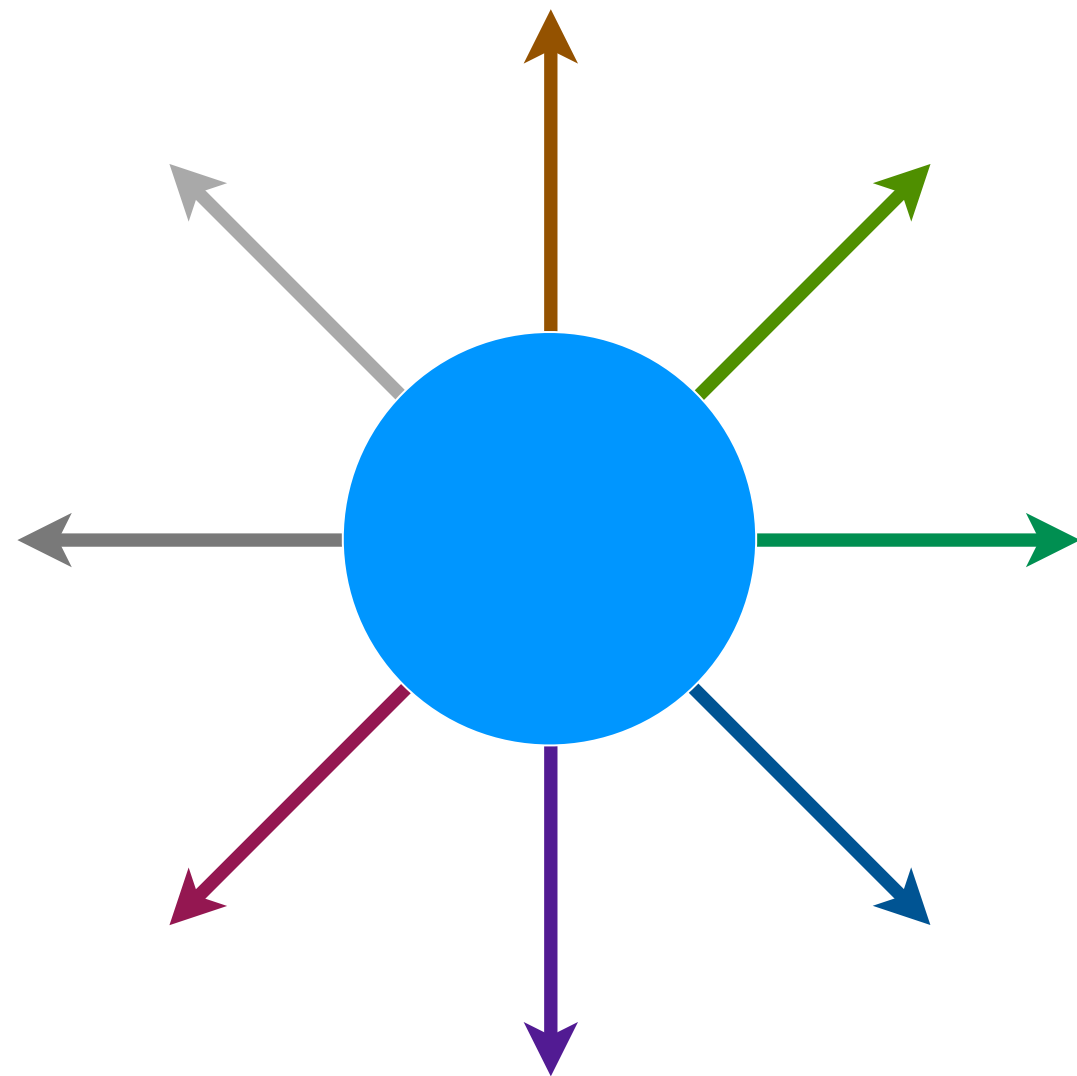
Context maps



Context maps



Context maps



Context controller

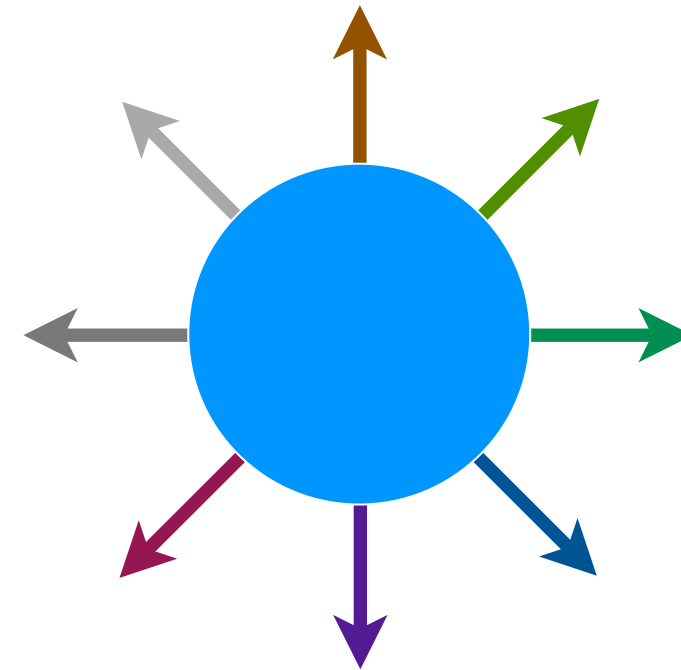
Danger

--	--	--	--	--	--	--	--

Interest

--	--	--	--	--	--	--	--

Plane behaviours



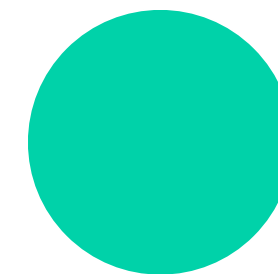
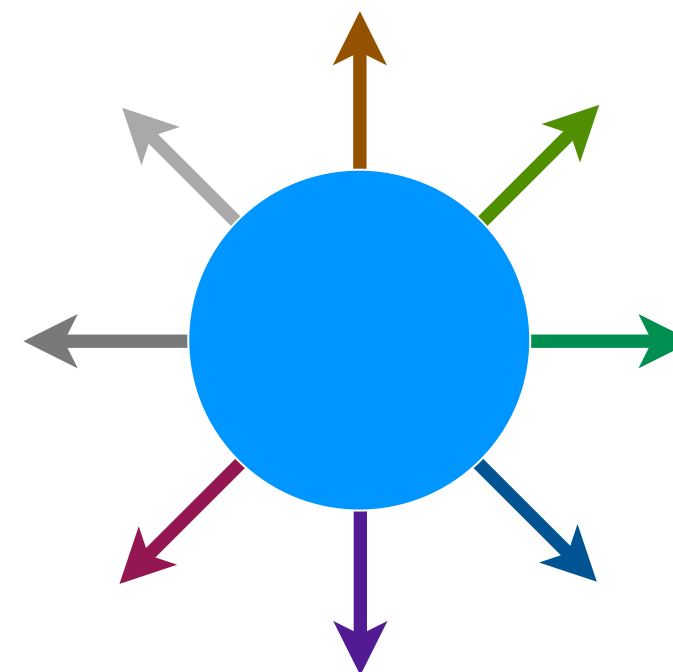
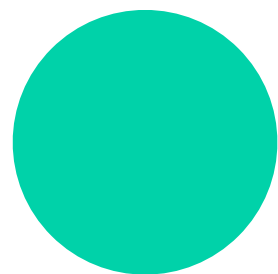
Danger

--	--	--	--	--	--	--	--

Interest

--	--	--	--	--	--	--	--

Chase behaviour



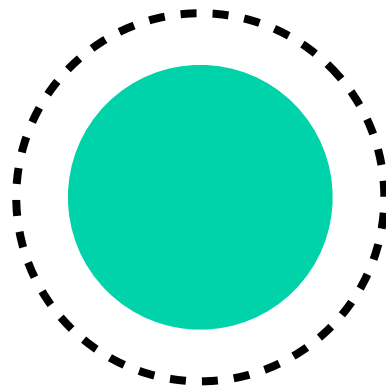
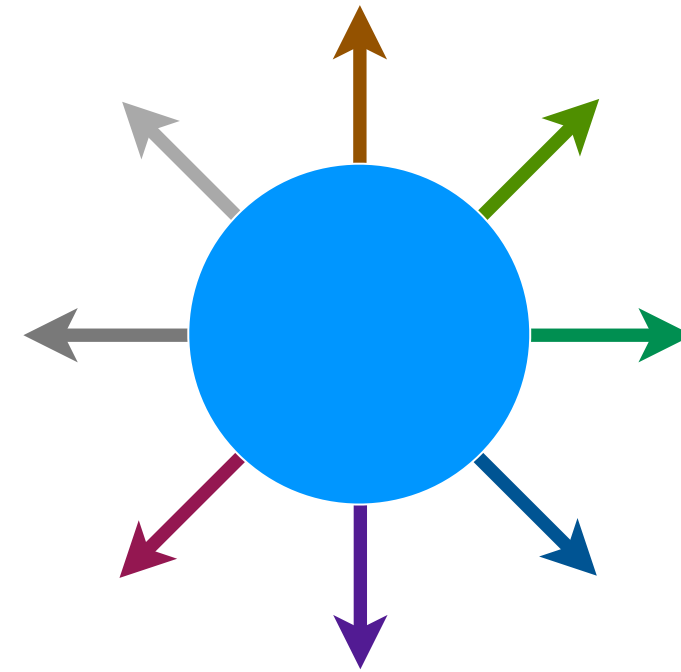
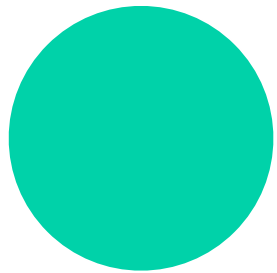
Danger

--	--	--	--	--	--	--	--

Interest

--	--	--	--	--	--	--	--

Chase behaviour



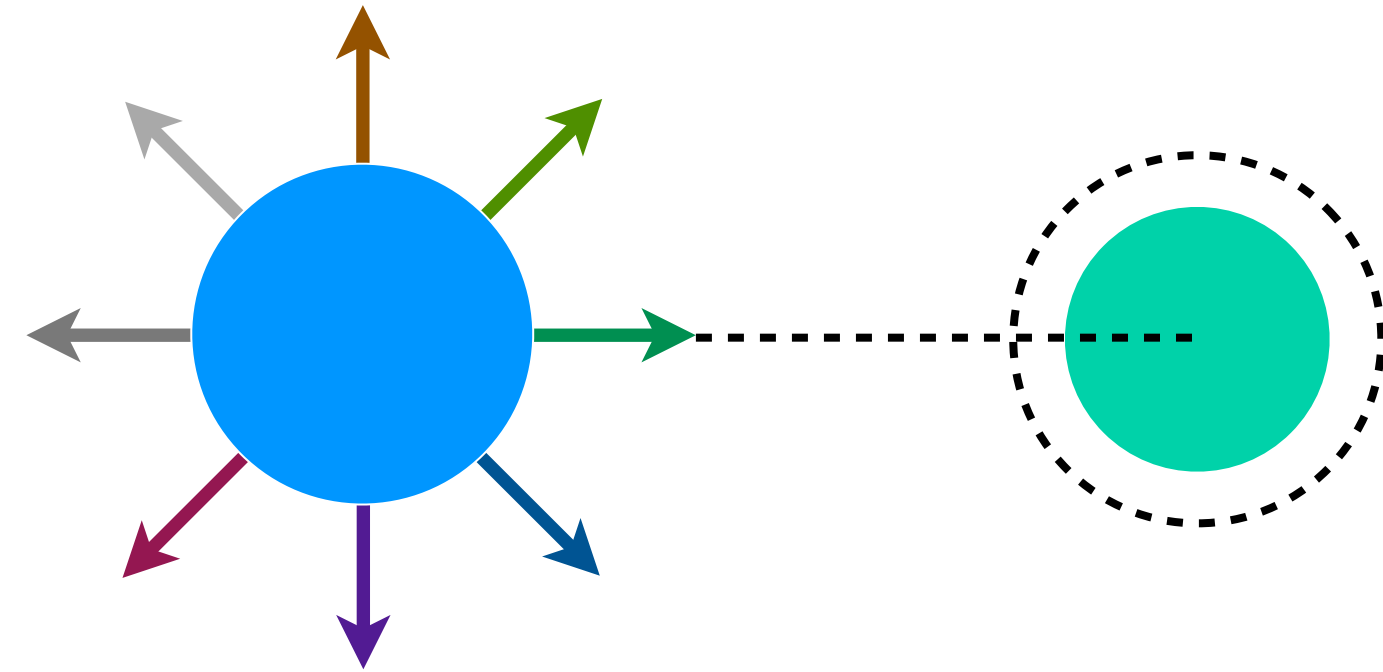
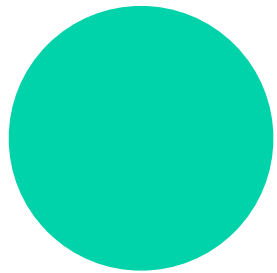
Danger

--	--	--	--	--	--	--	--

Interest

--	--	--	--	--	--	--	--

Chase behaviour



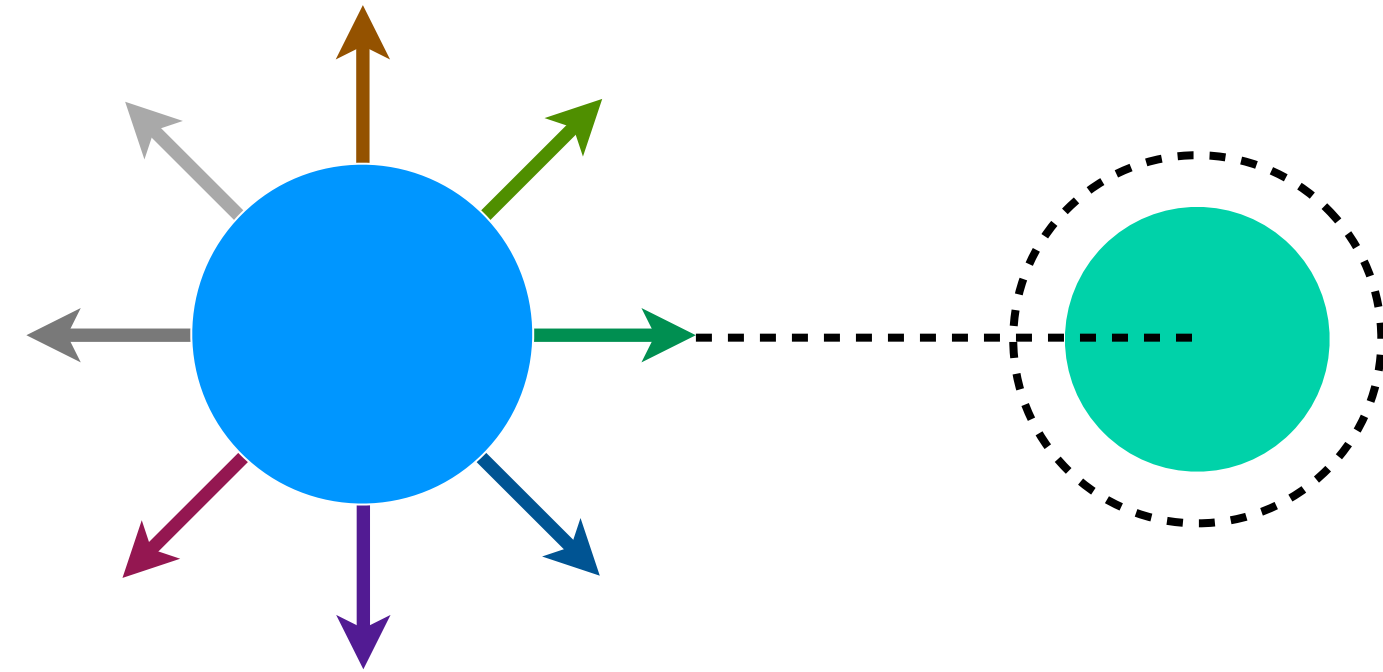
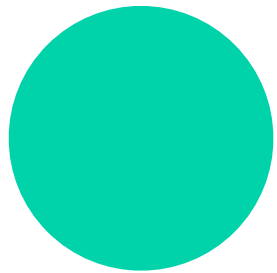
Danger

--	--	--	--	--	--	--	--

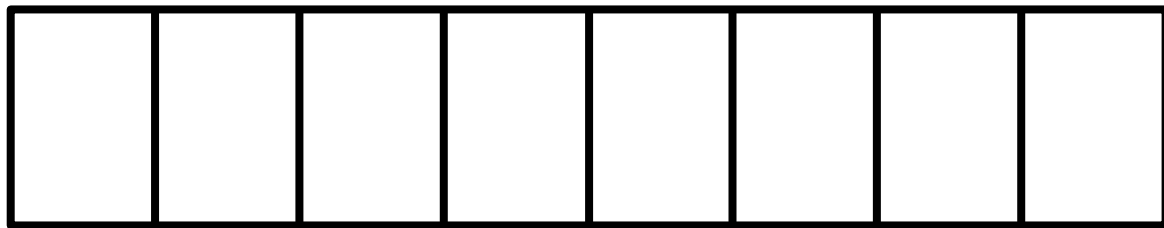
Interest

--	--	--	--	--	--	--	--

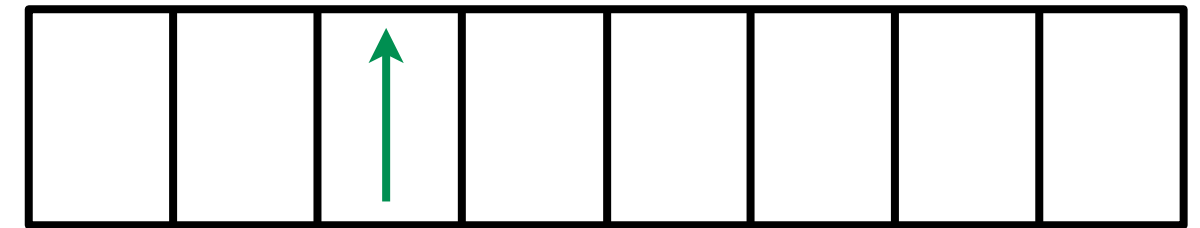
Chase behaviour



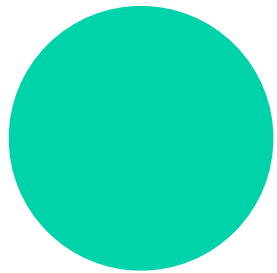
Danger



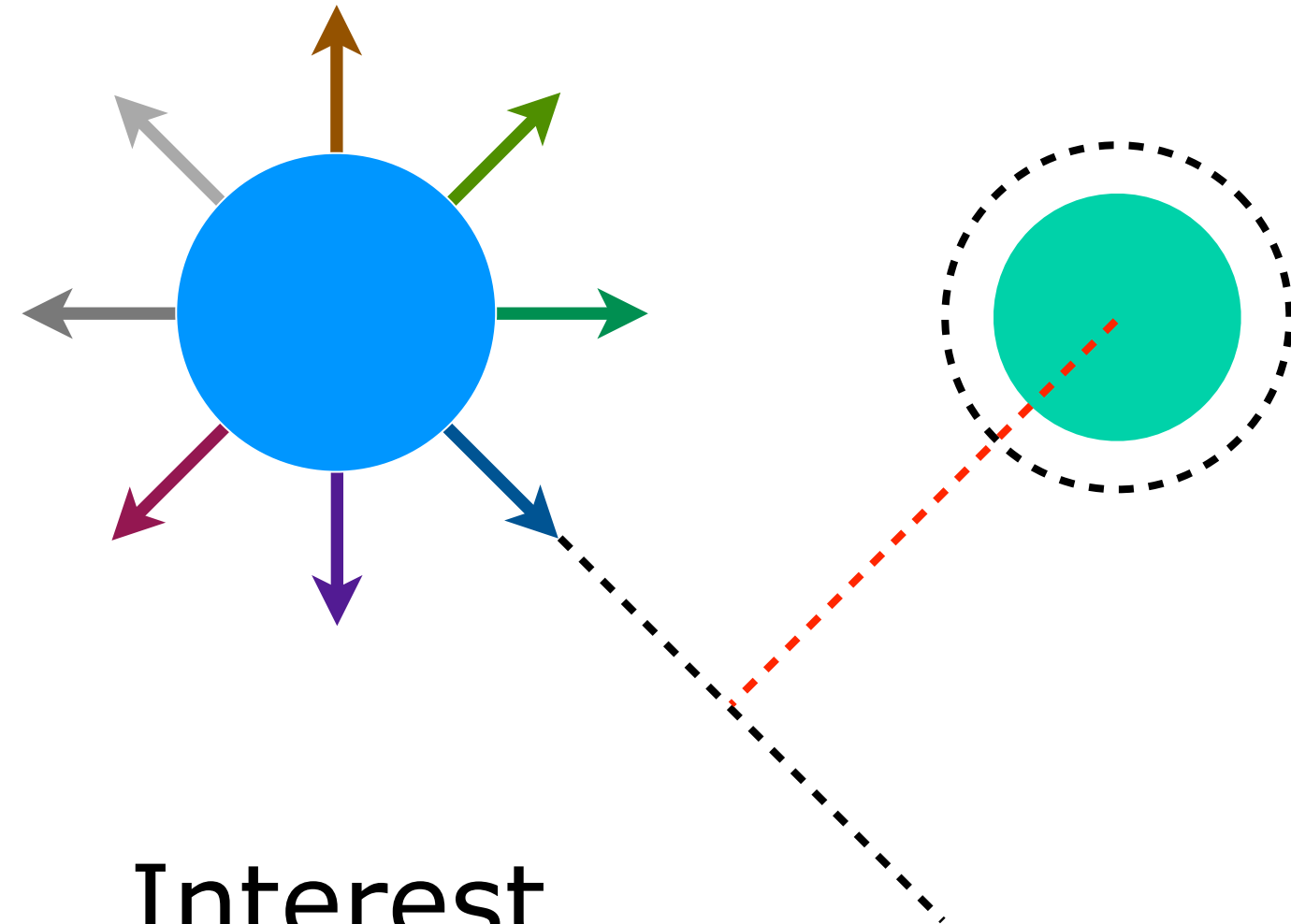
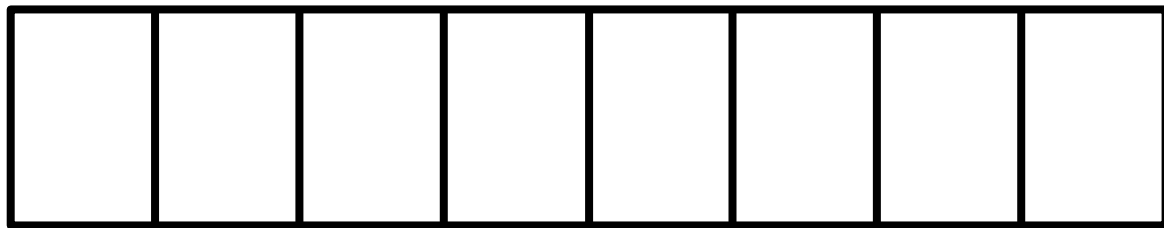
Interest



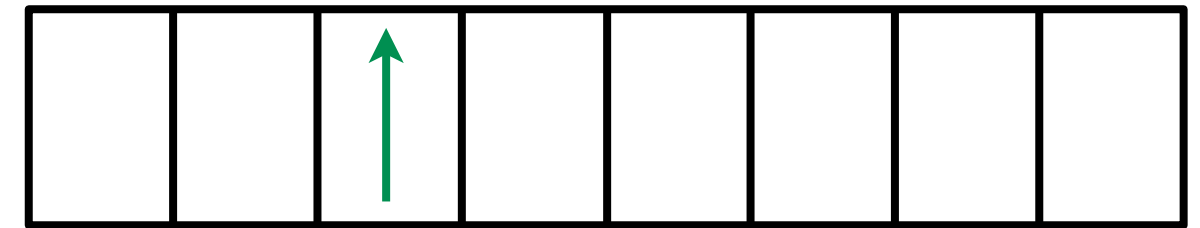
Chase behaviour



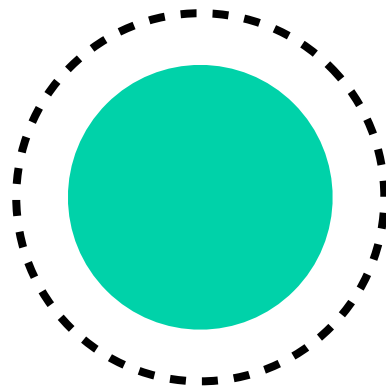
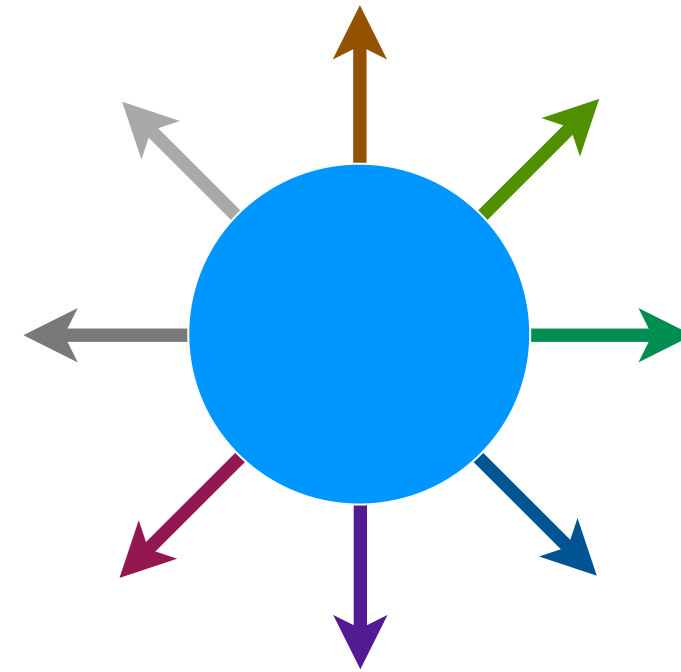
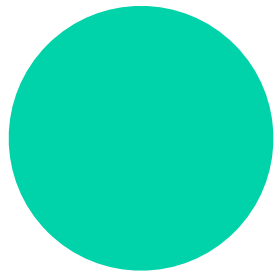
Danger



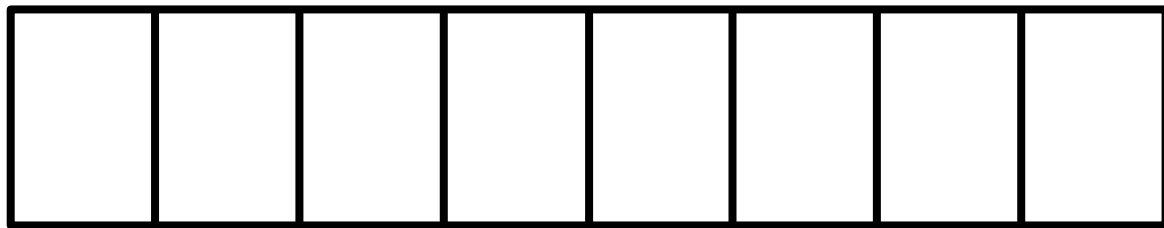
Interest



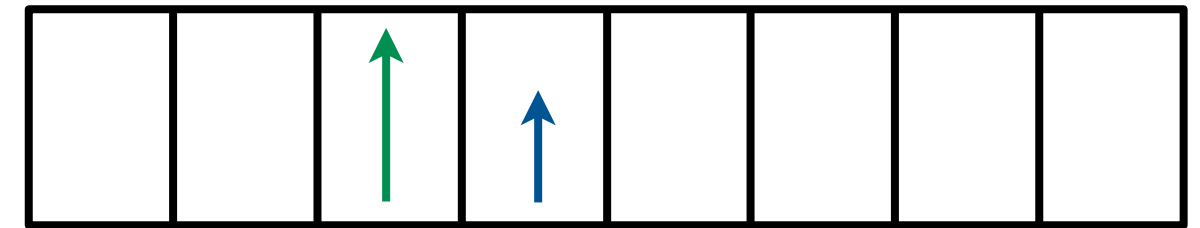
Chase behaviour



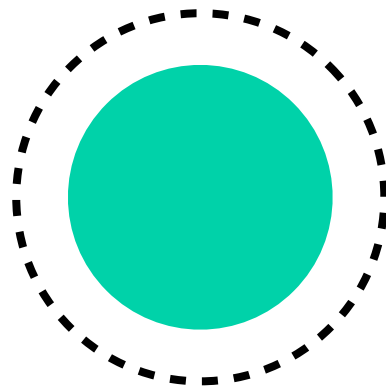
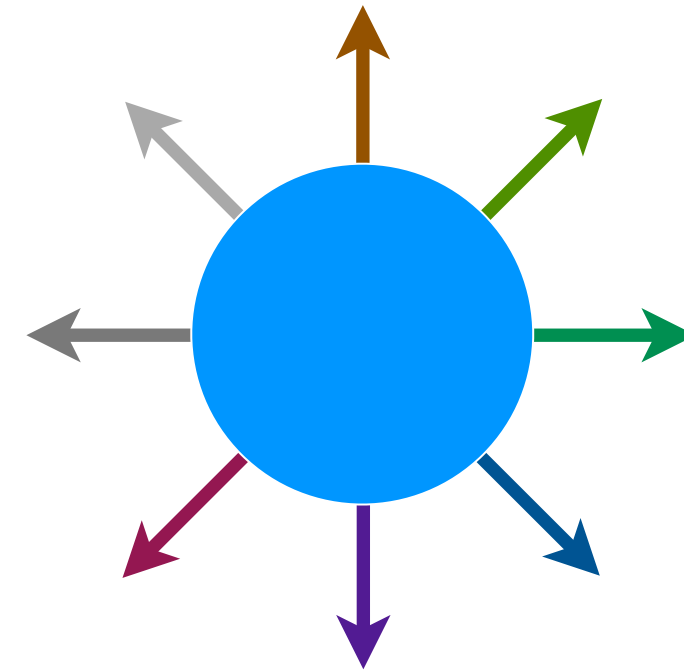
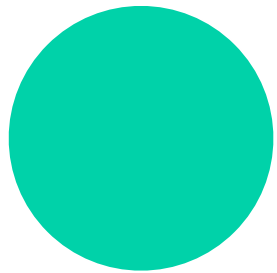
Danger



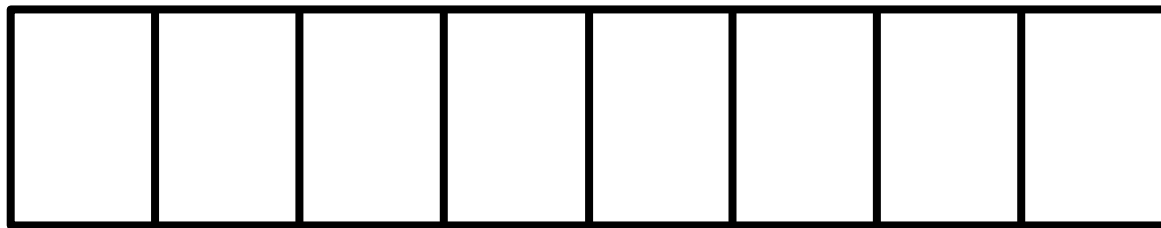
Interest



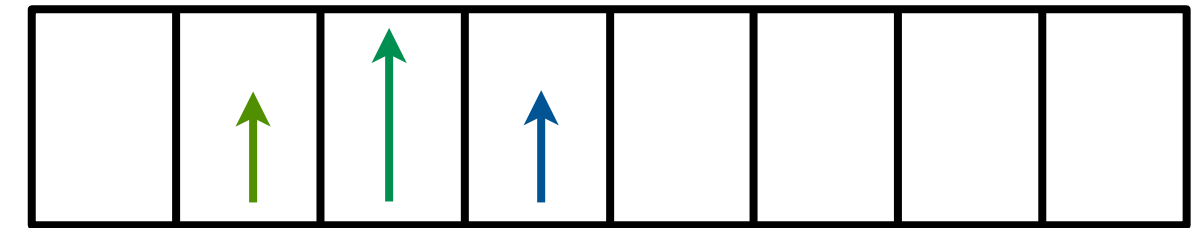
Chase behaviour



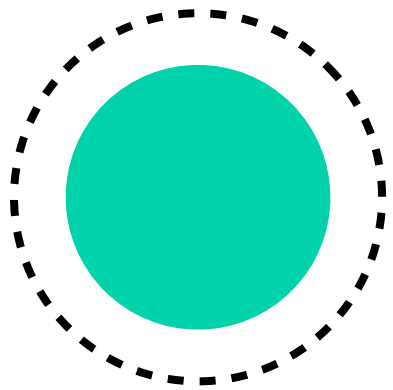
Danger



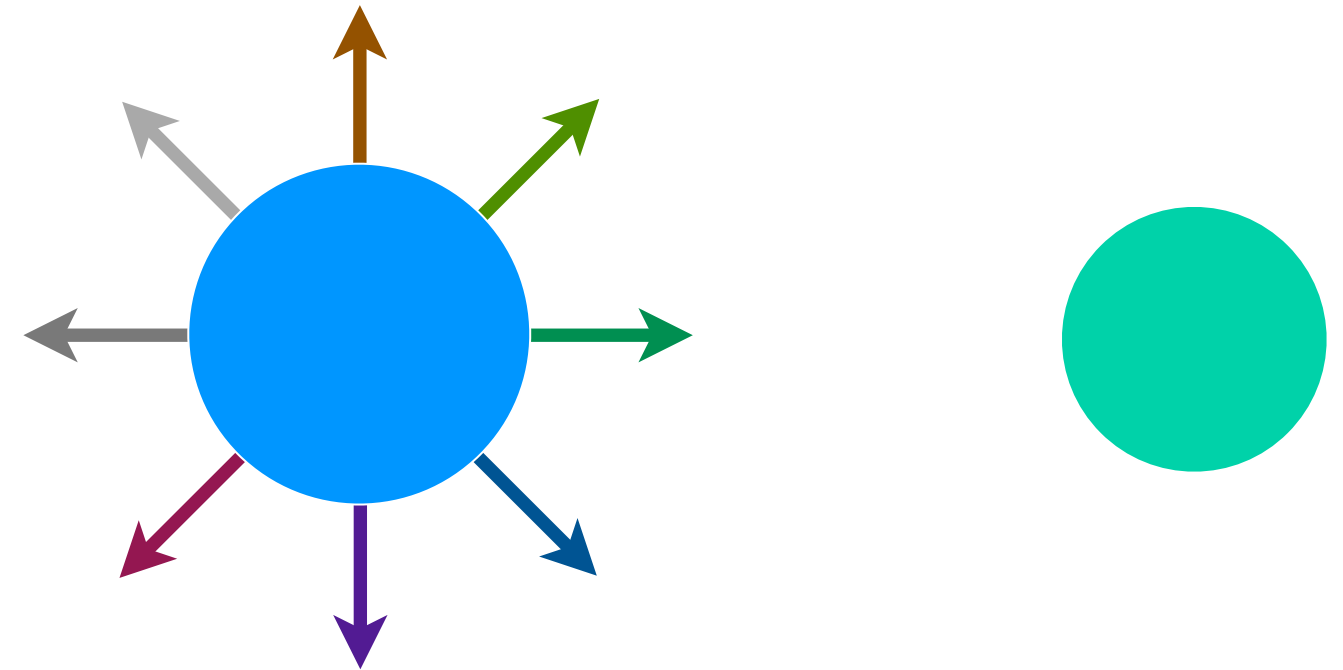
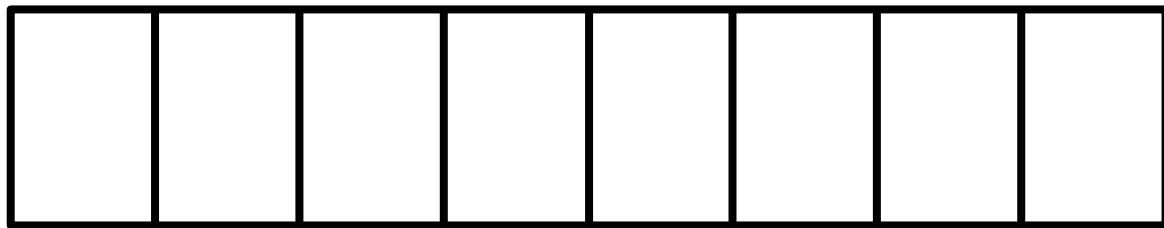
Interest



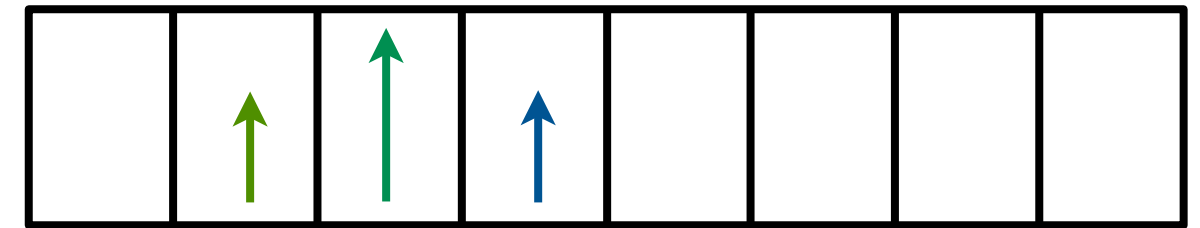
Chase behaviour



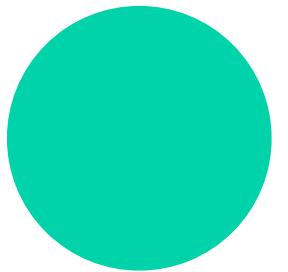
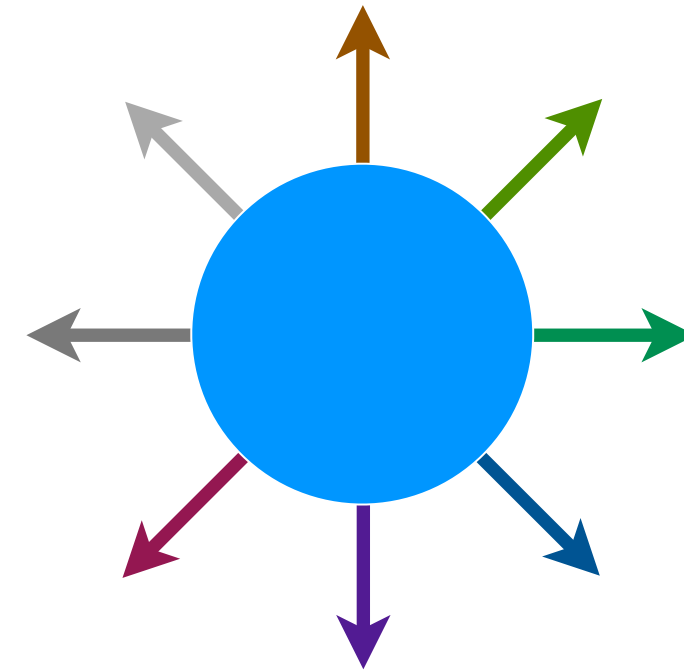
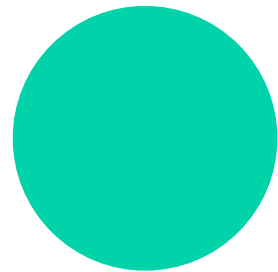
Danger



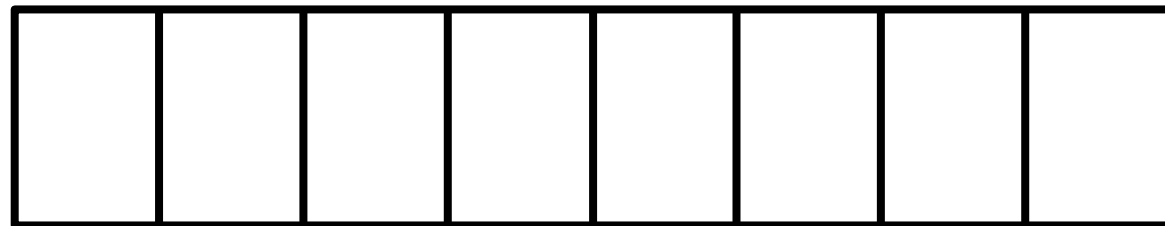
Interest



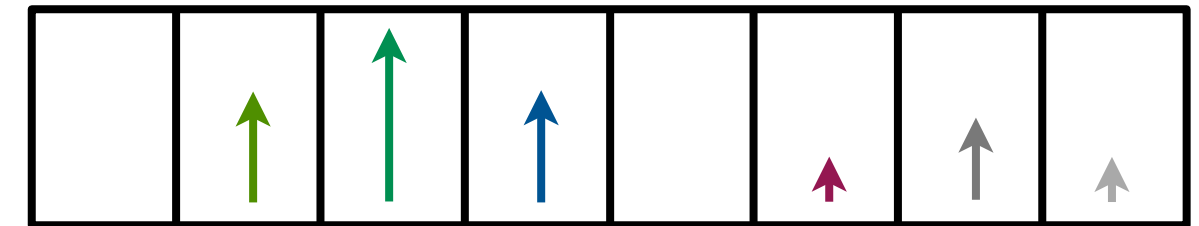
Chase behaviour



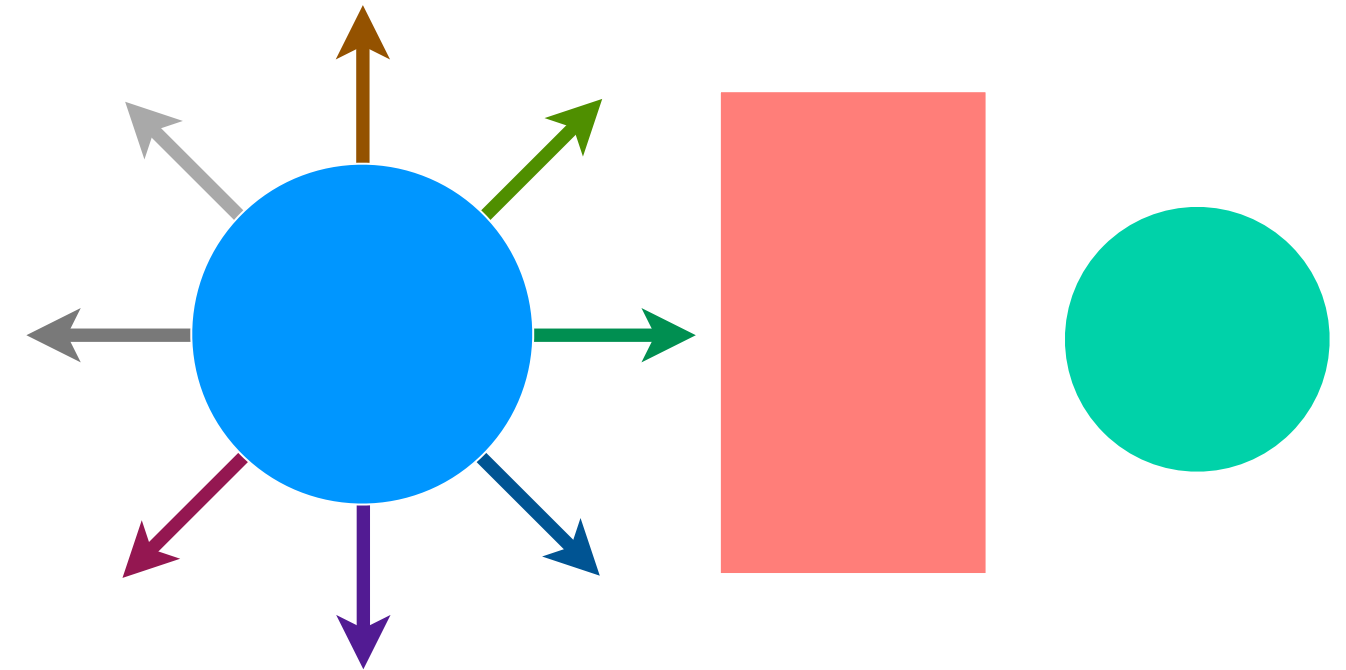
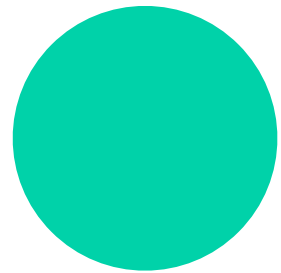
Danger



Interest



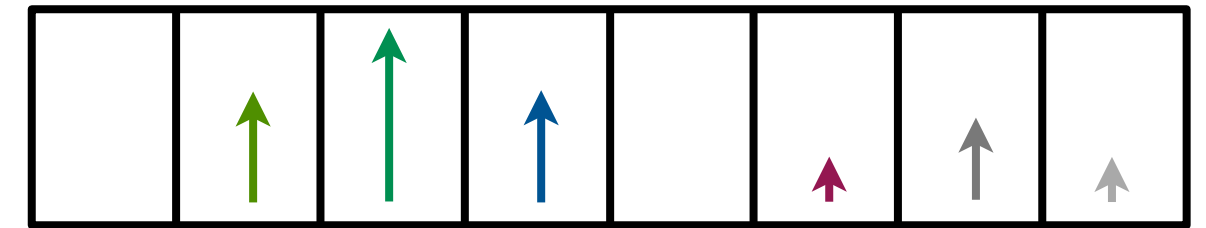
Avoid behaviour



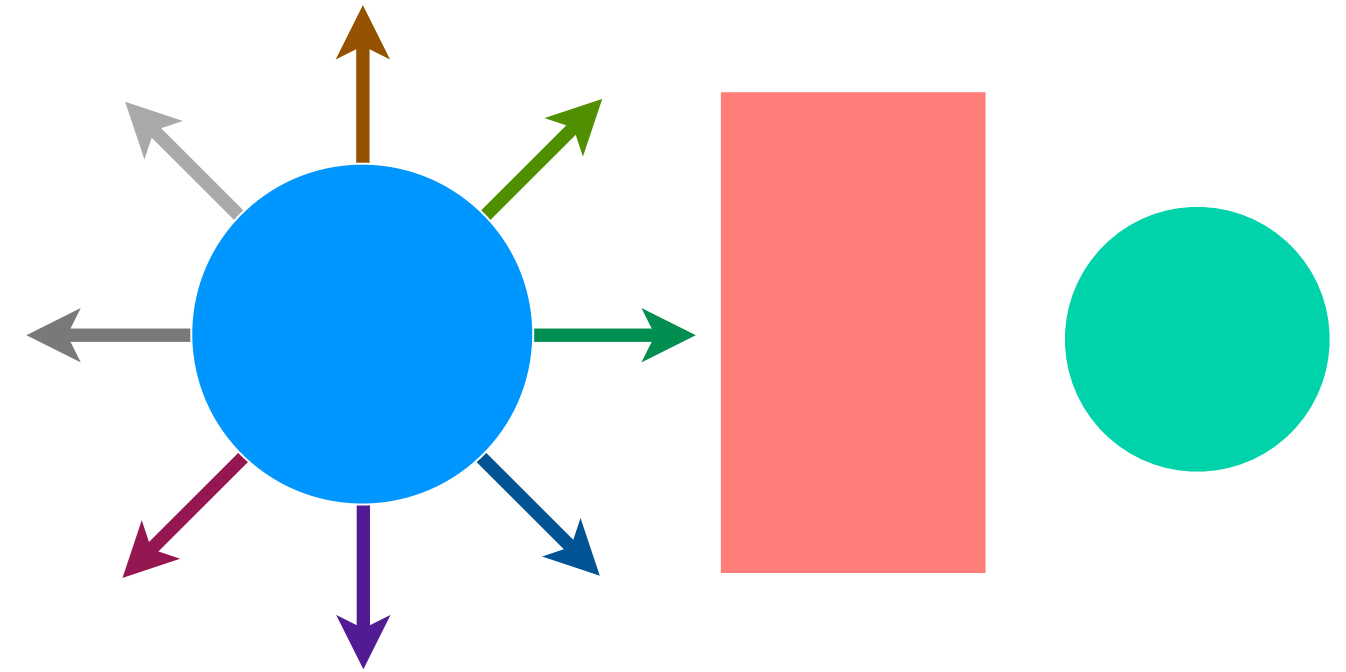
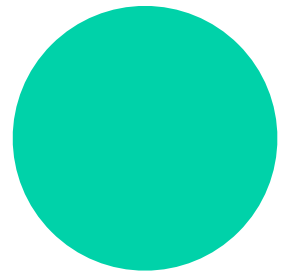
Danger



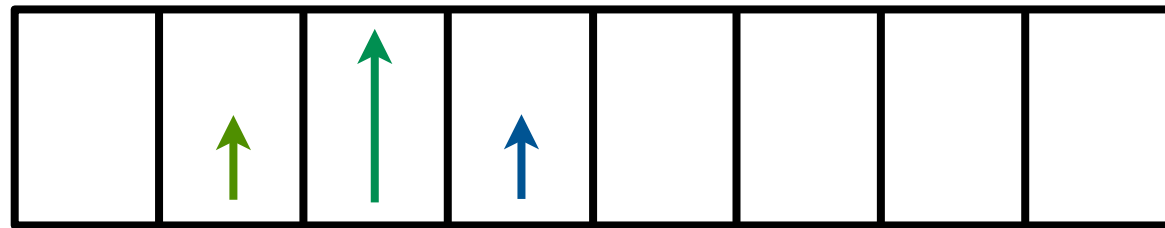
Interest



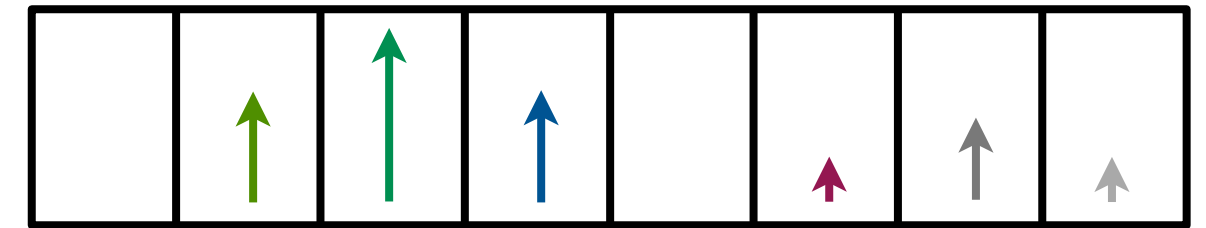
Avoid behaviour



Danger

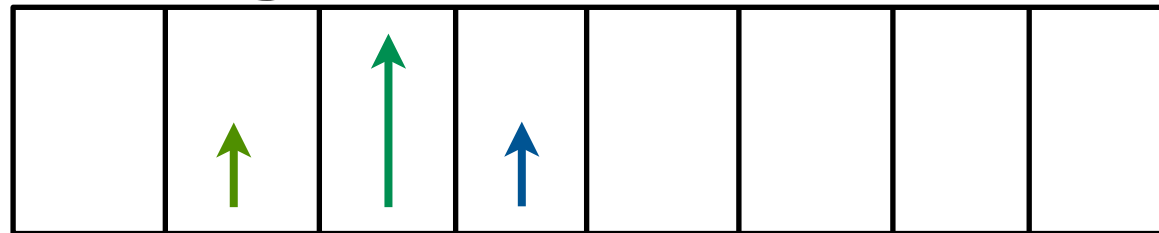


Interest

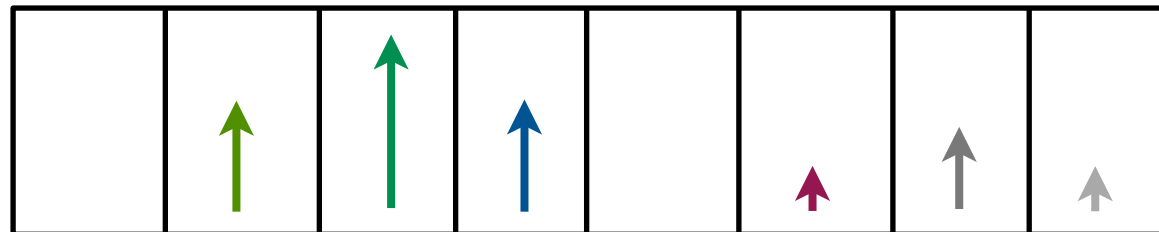


Processing the maps

Danger

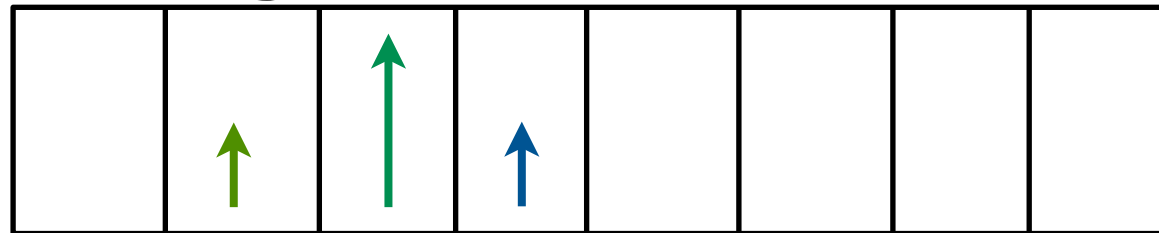


Interest



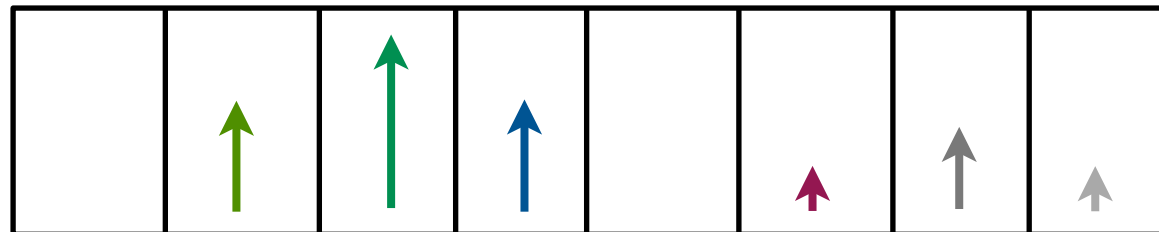
Processing the maps

Danger



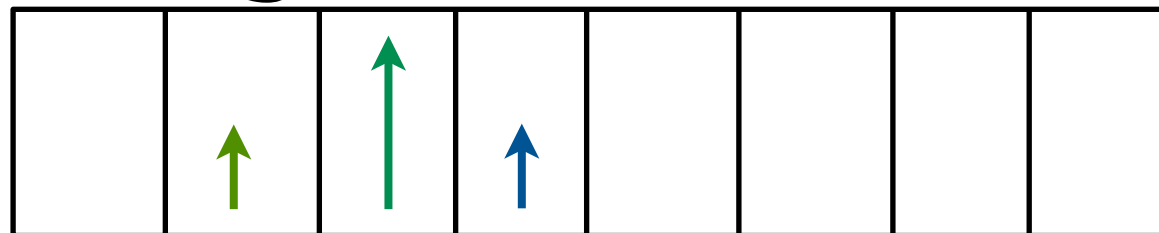
X X X

Interest



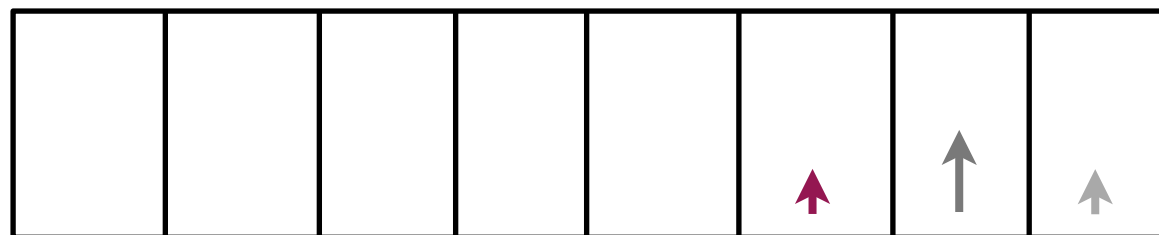
Processing the maps

Danger



X X X

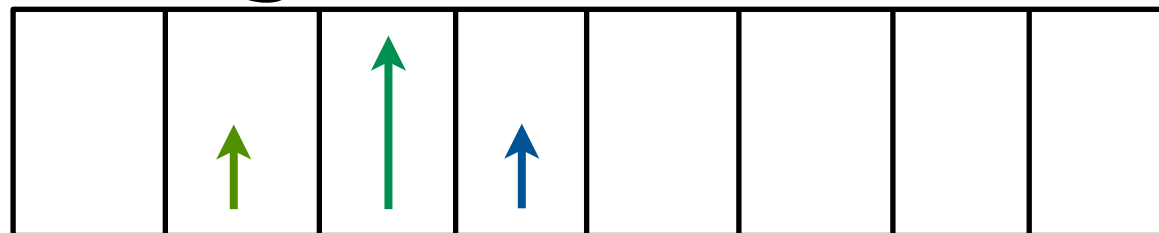
Interest



X X X

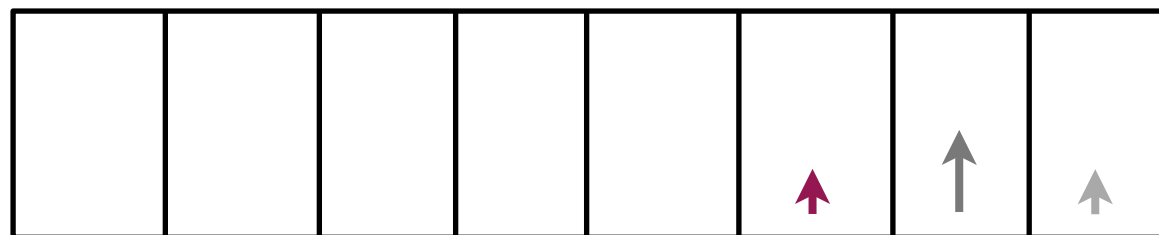
Processing the maps

Danger



X X X

Interest

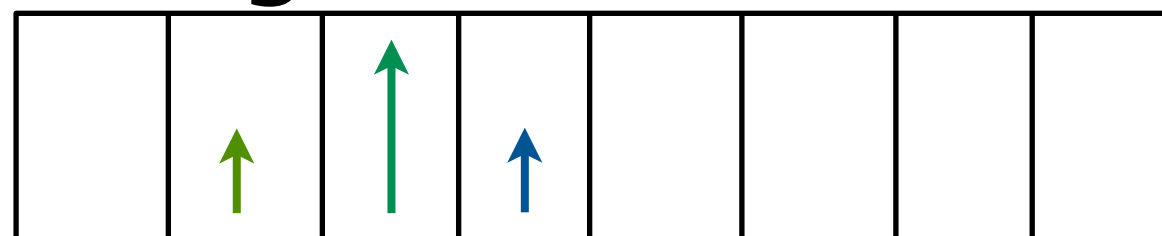


X X X



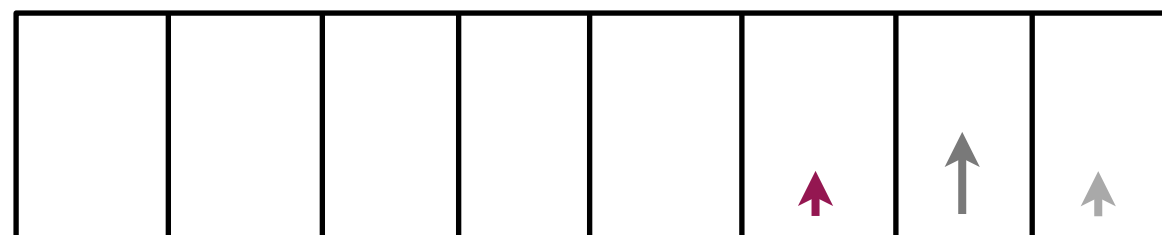
Processing the maps

Danger

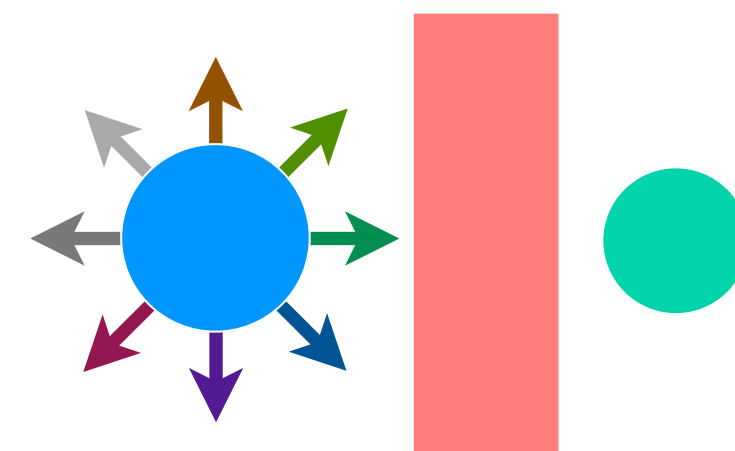
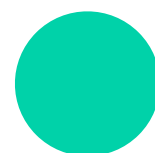


X X X

Interest

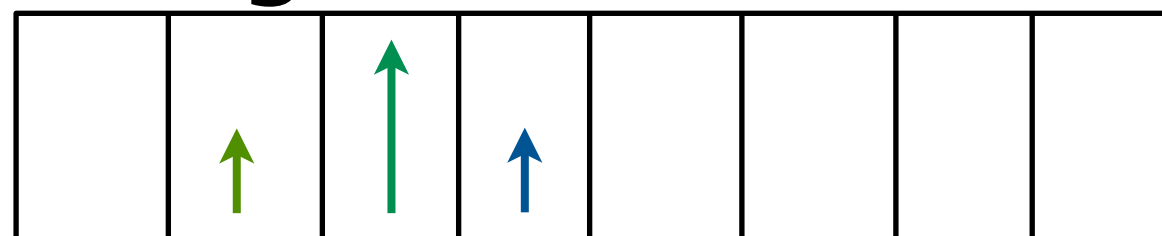


X X X



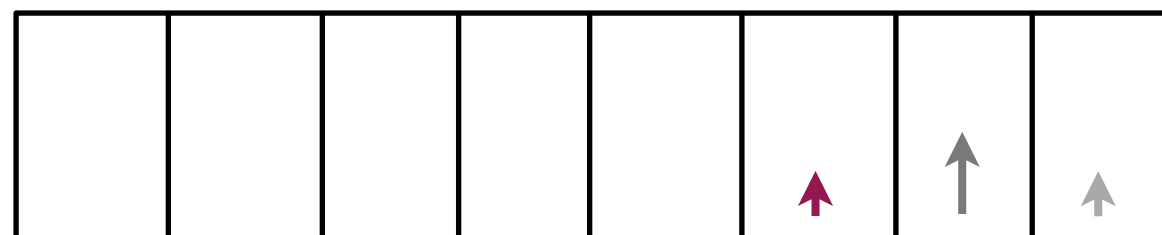
Processing the maps

Danger

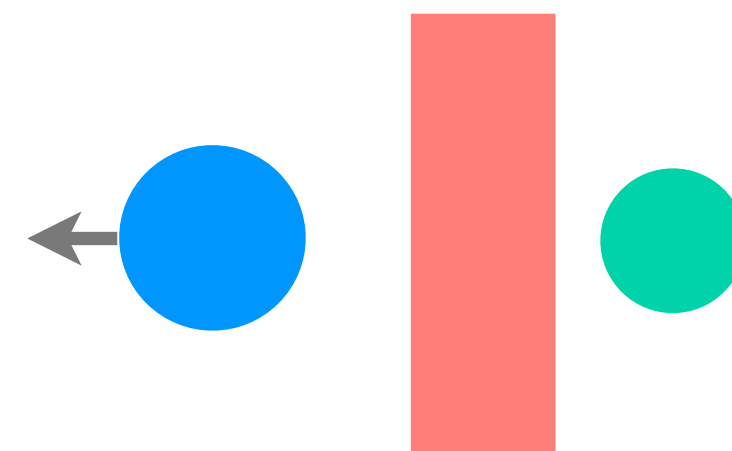
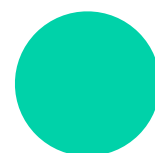


X X X

Interest

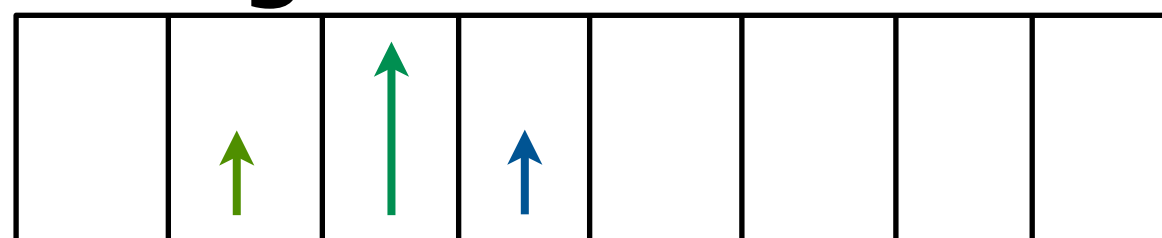


X X X



Processing the maps

Danger

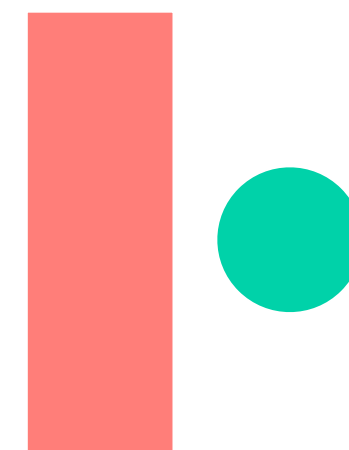
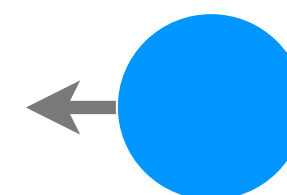
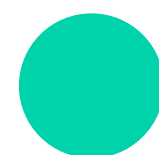


X X X

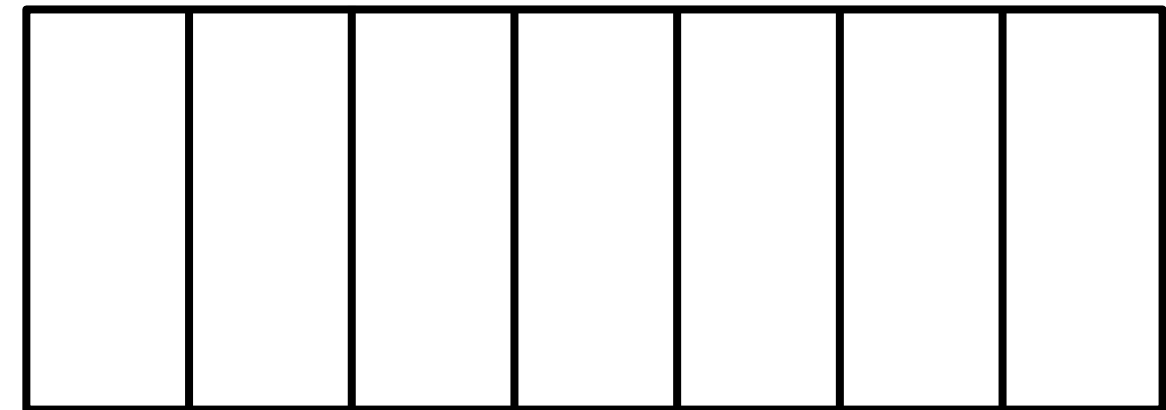
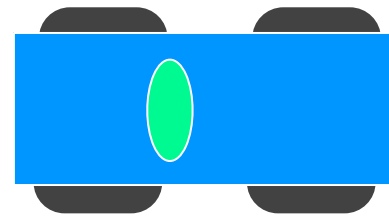
Interest



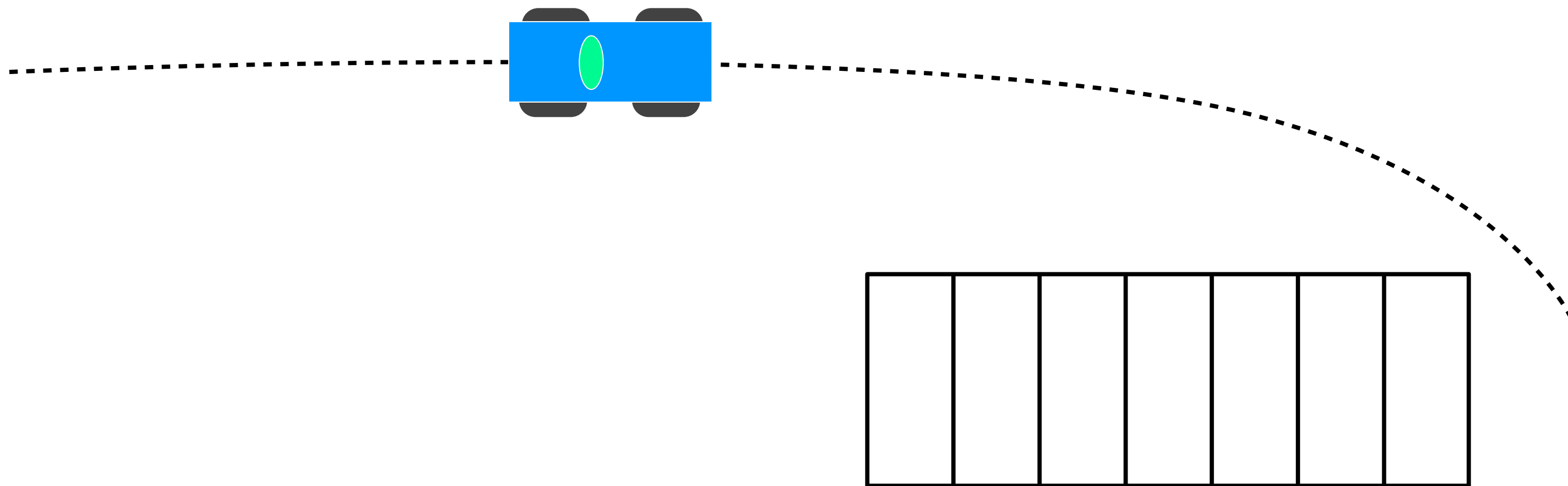
X X X



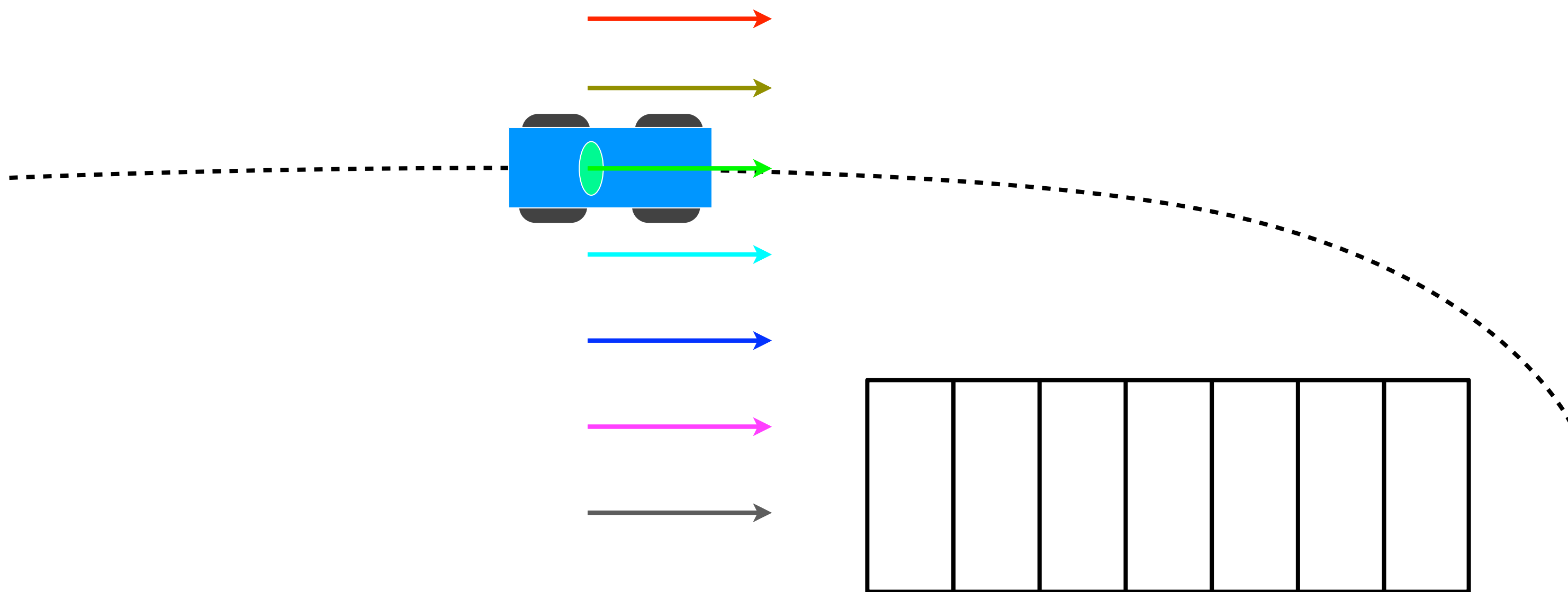
Racing context map projection



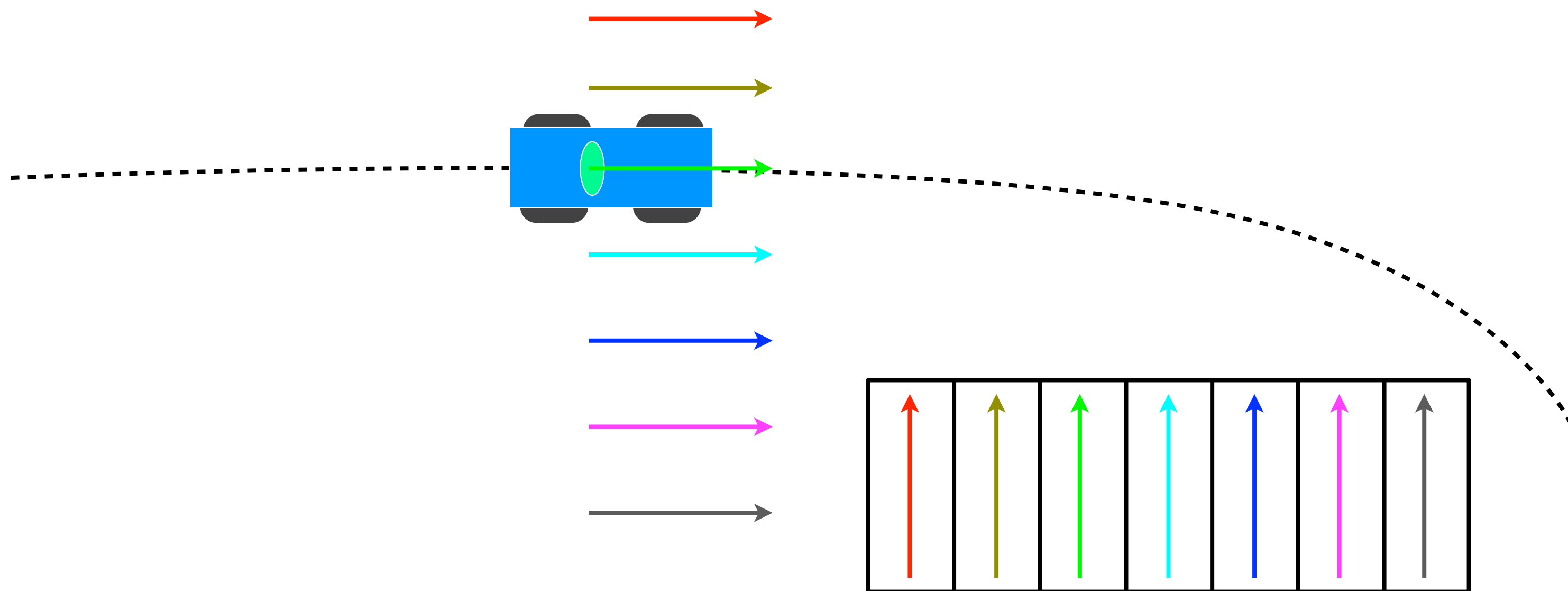
Racing context map projection



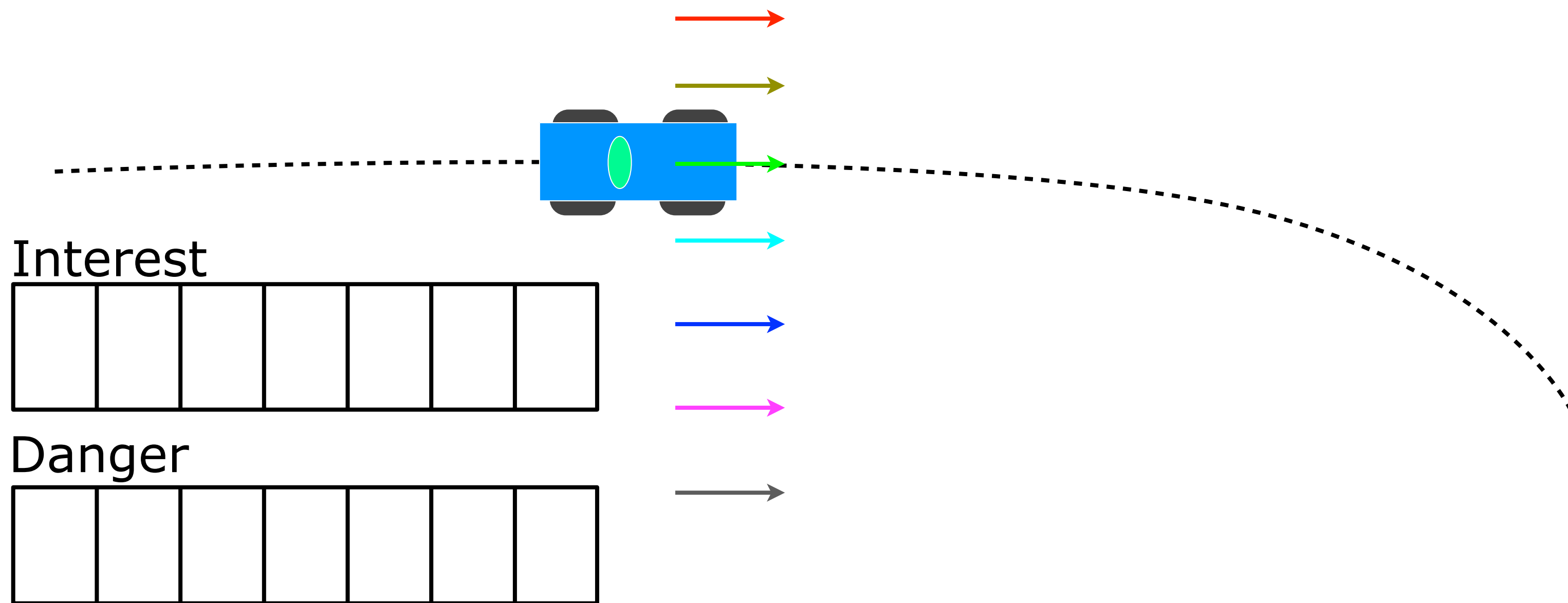
Racing context map projection



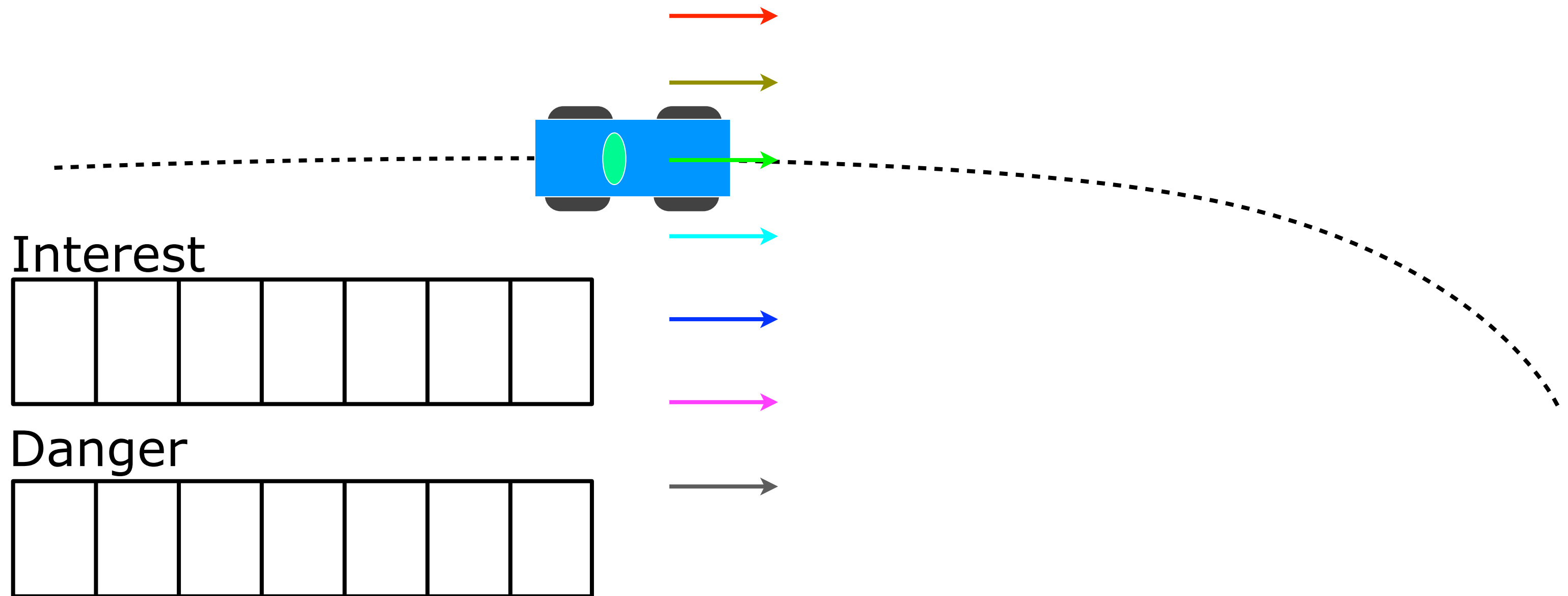
Racing context map projection



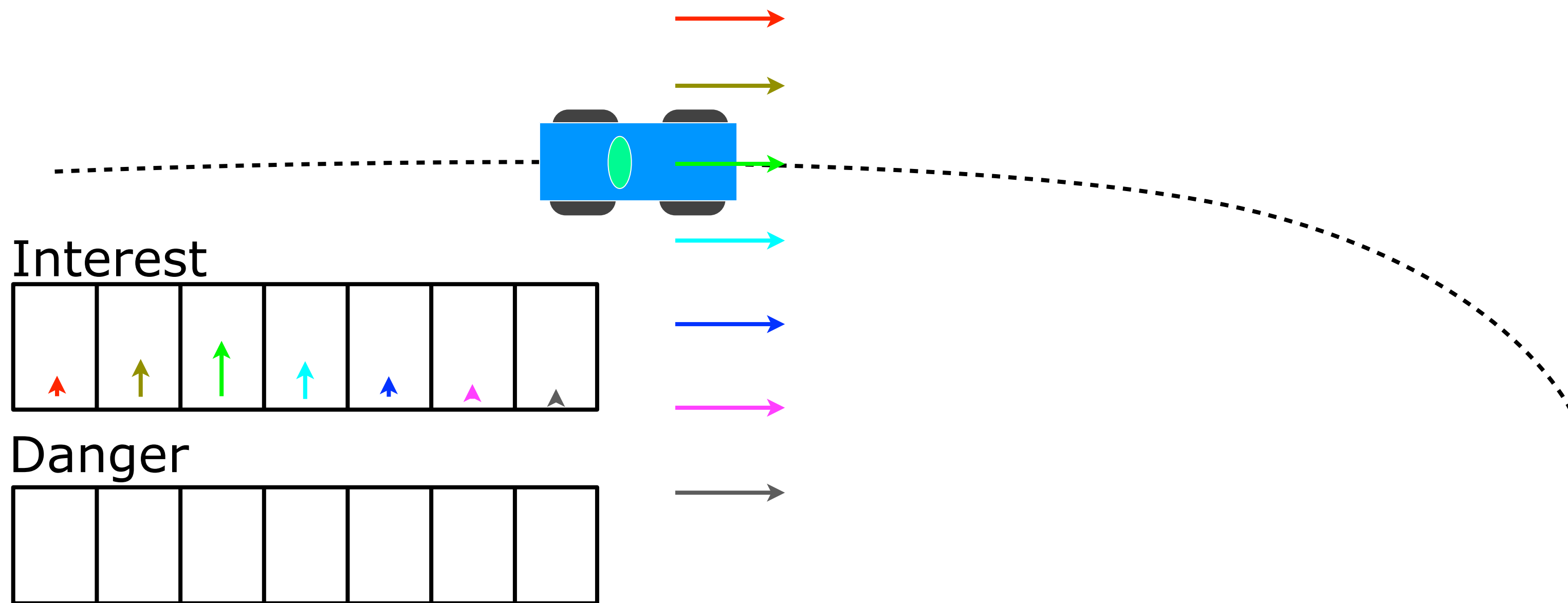
Racing behaviours



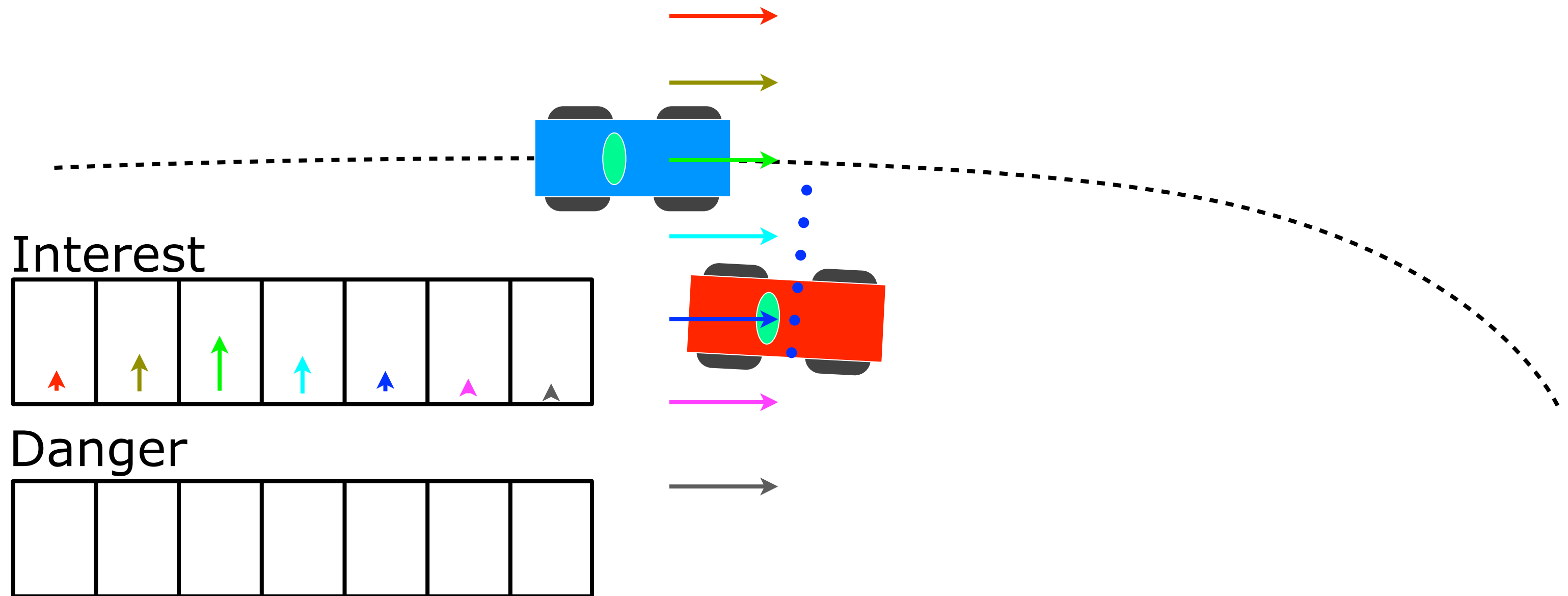
Racing line behaviour



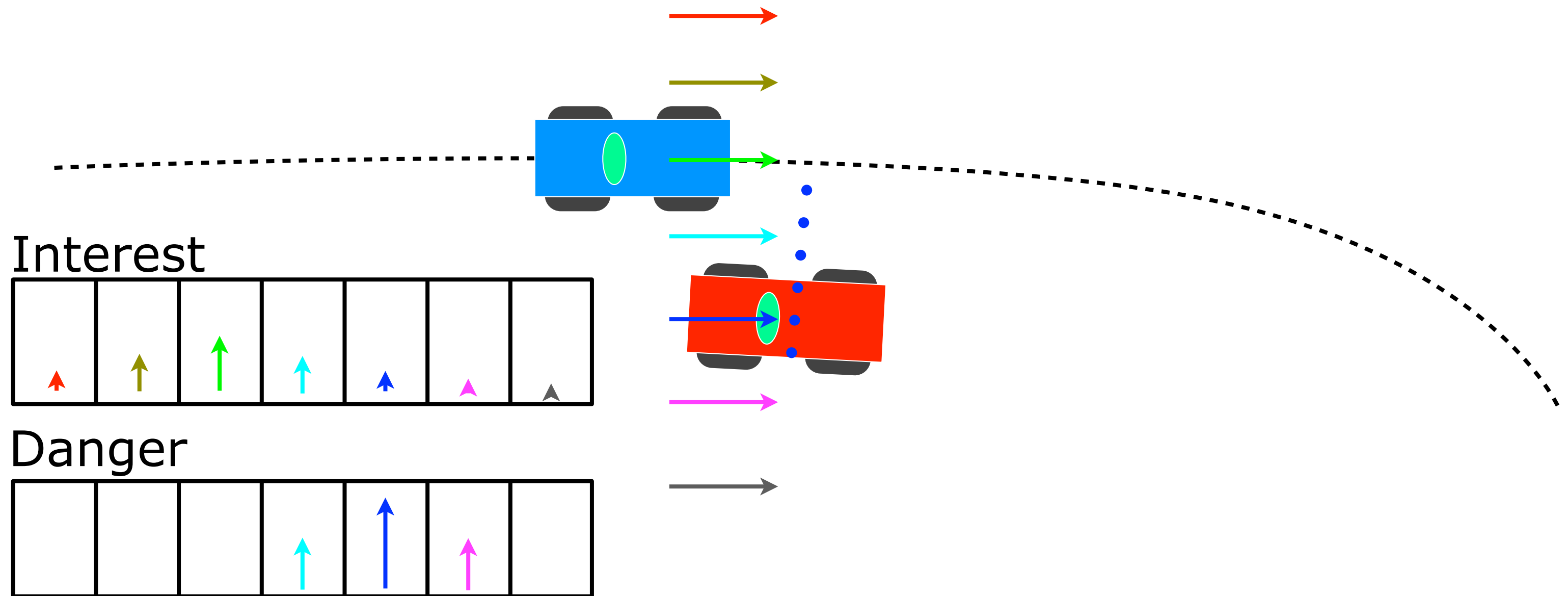
Racing line behaviour



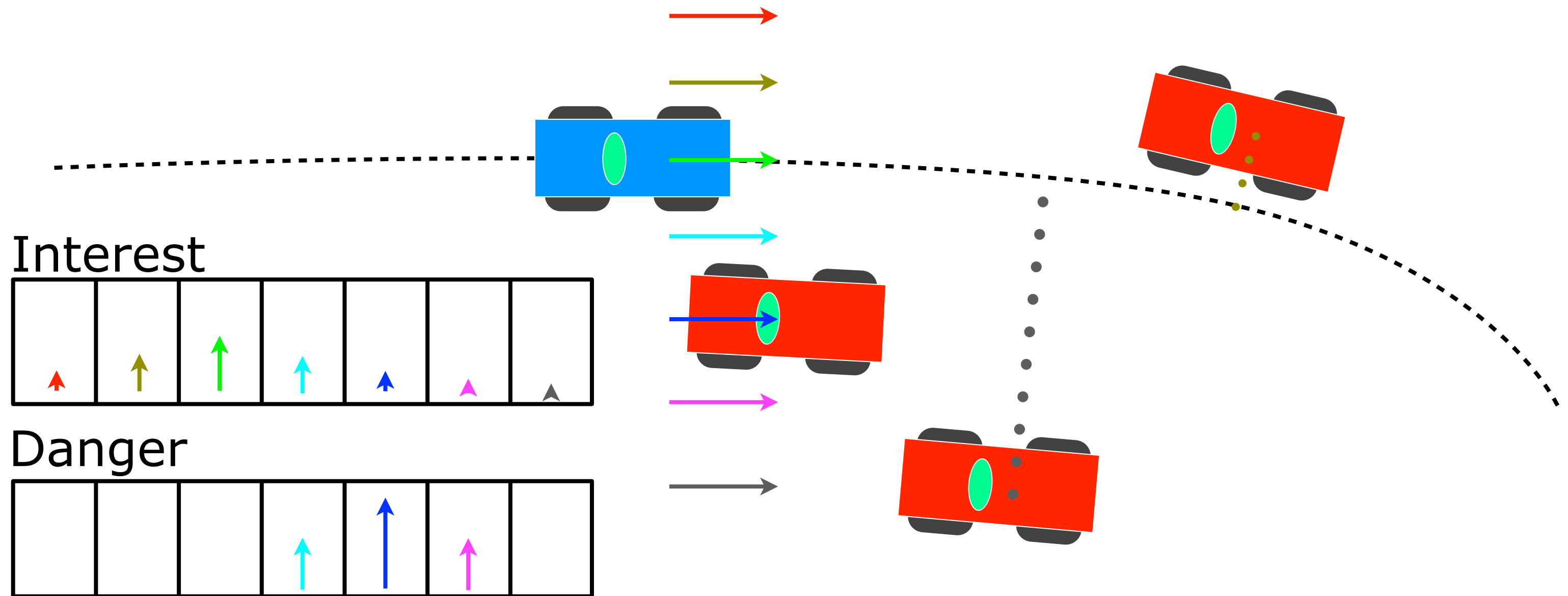
Avoid behaviour



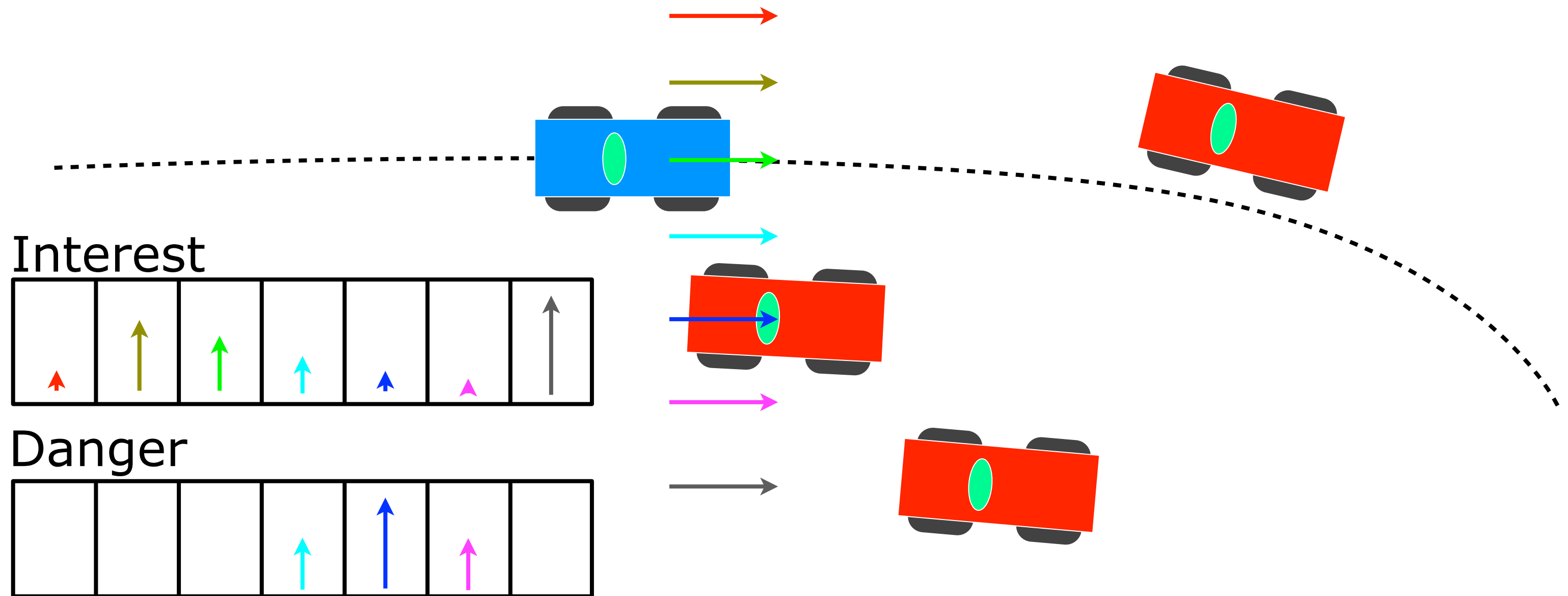
Avoid behaviour



Draft behaviour

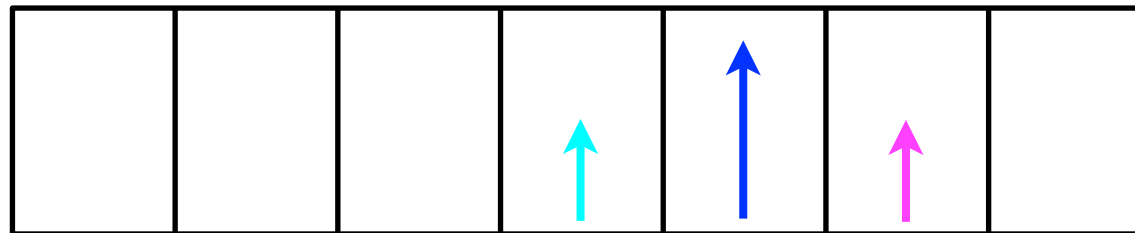


Draft behaviour

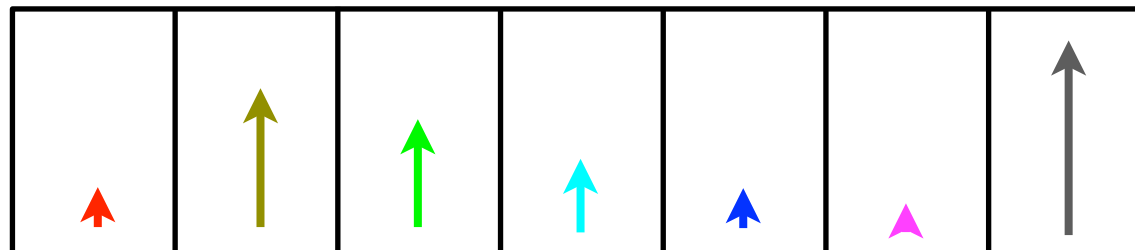


Processing the maps

Danger

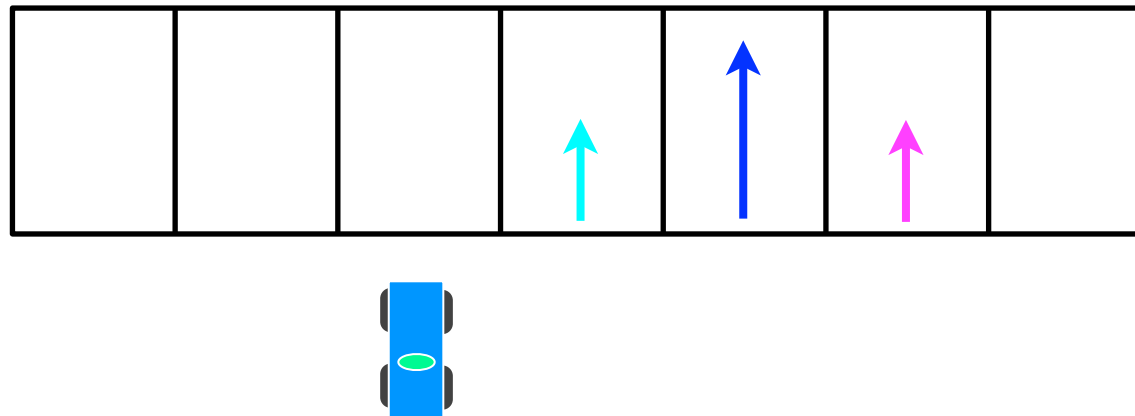


Interest

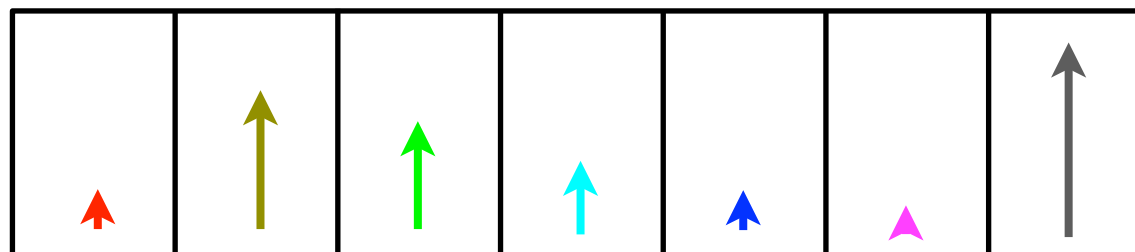


Processing the maps

Danger

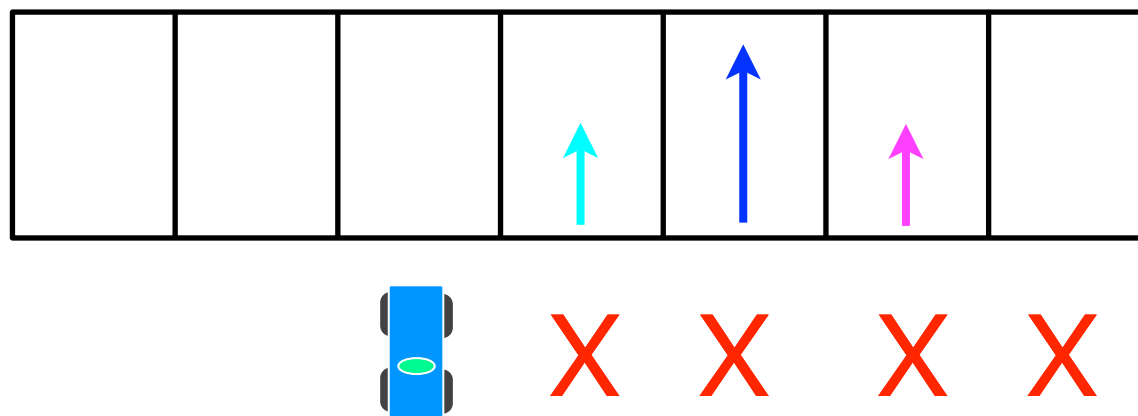


Interest

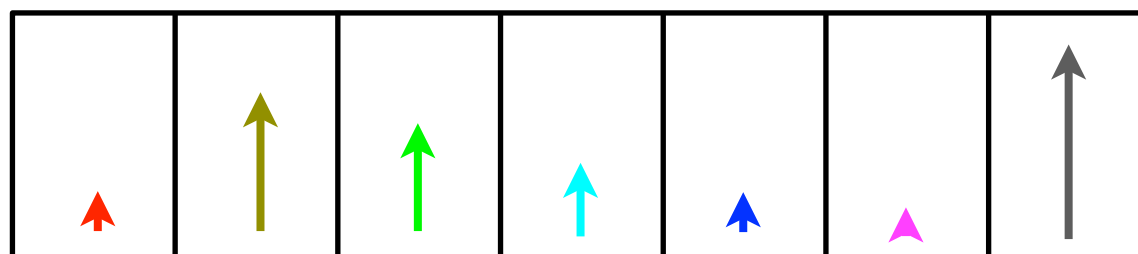


Processing the maps

Danger

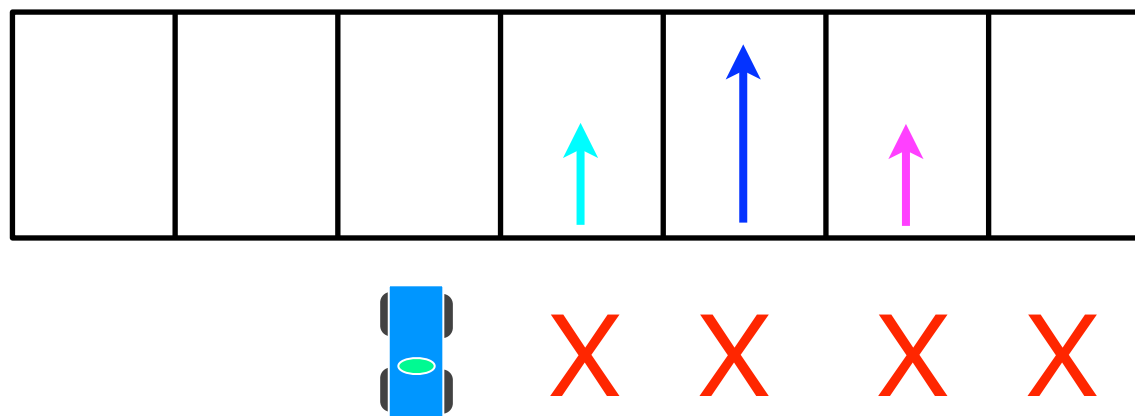


Interest

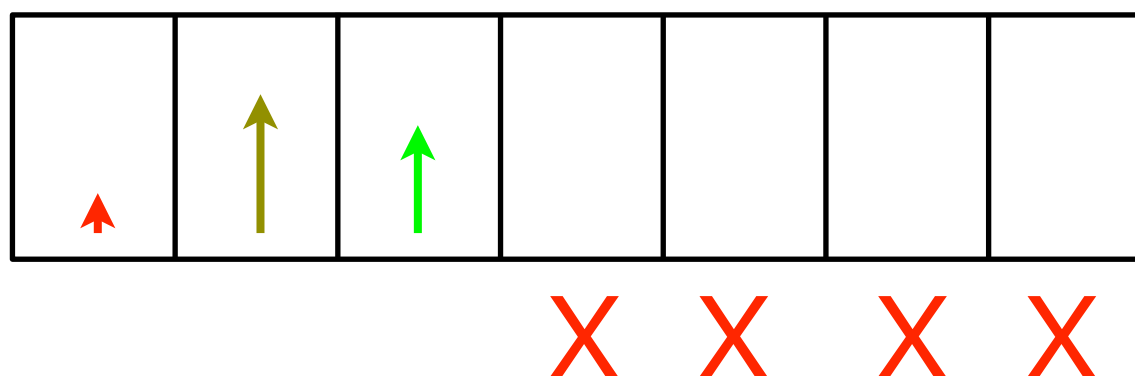


Processing the maps

Danger

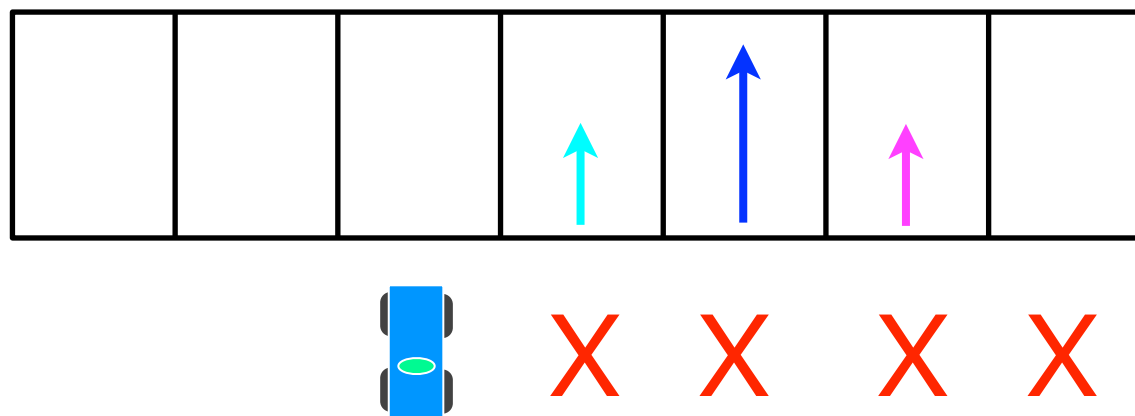


Interest

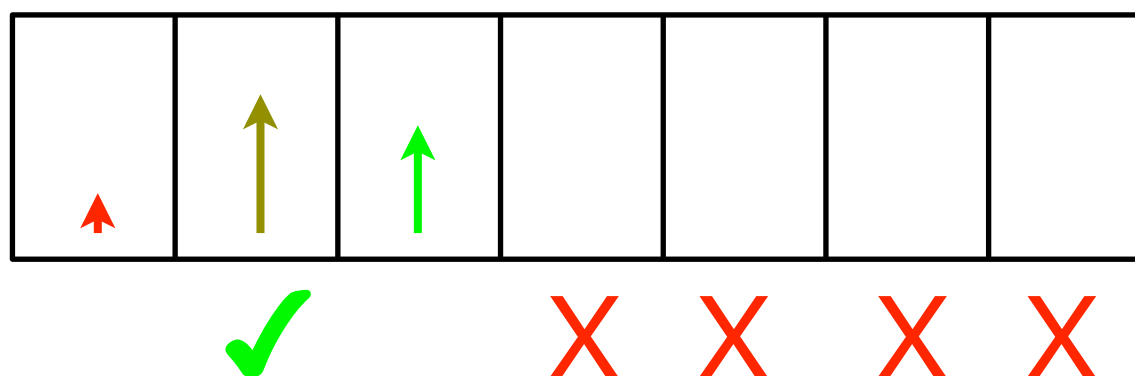


Processing the maps

Danger

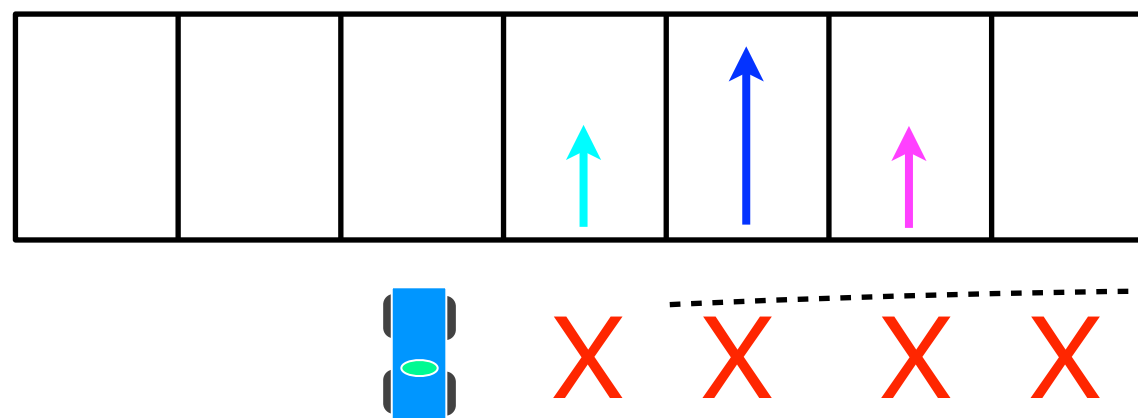


Interest

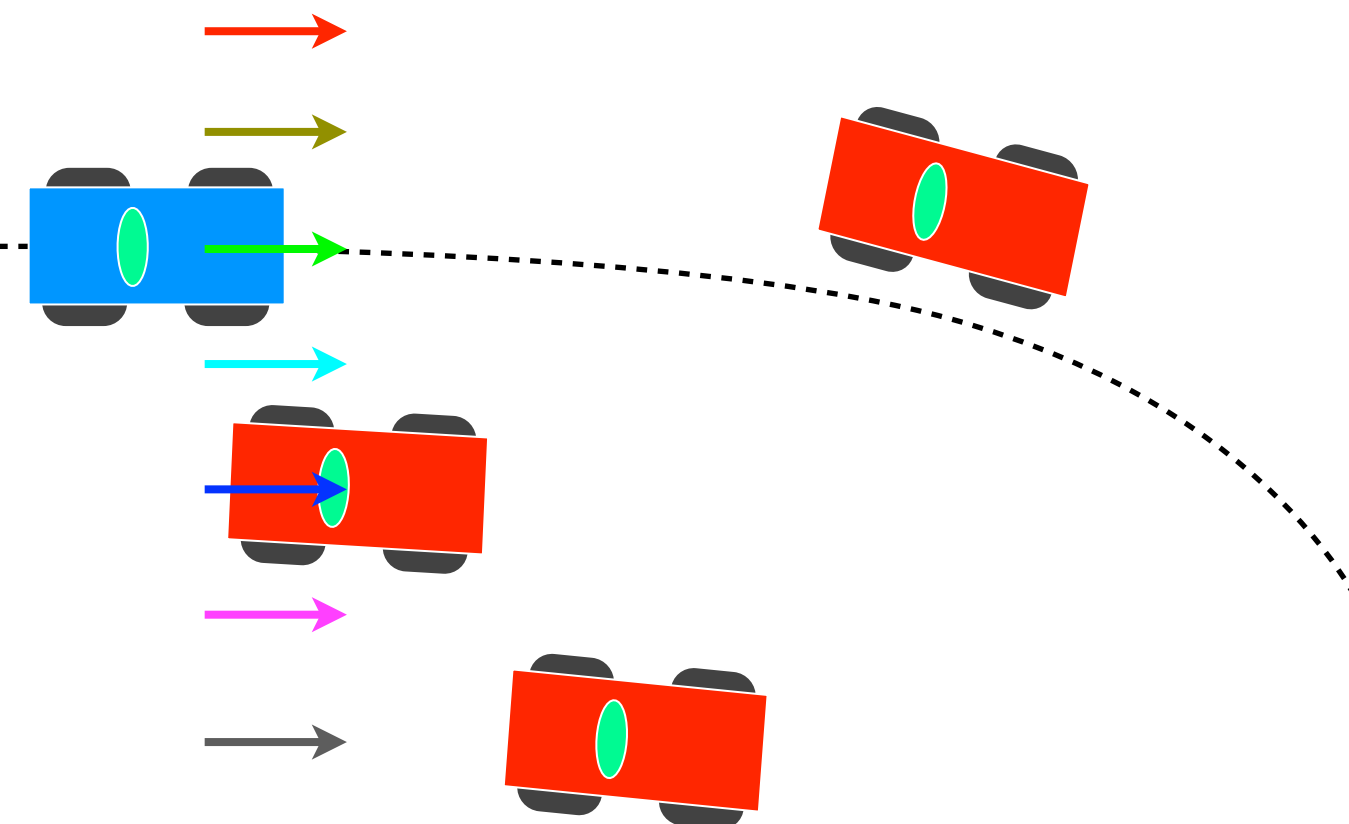
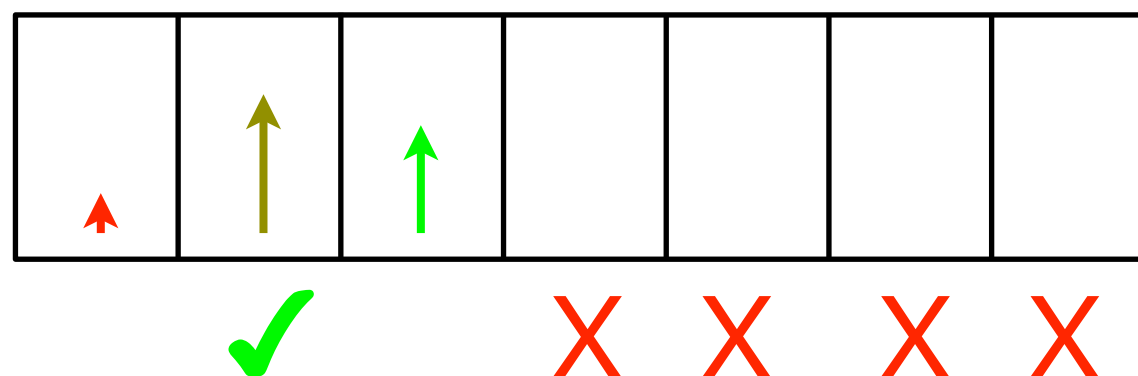


Processing the maps

Danger

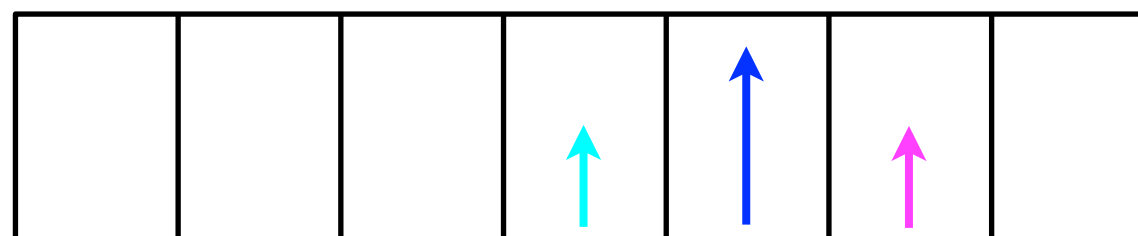


Interest

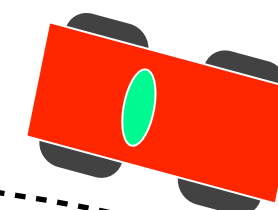
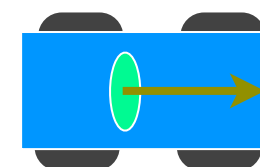


Processing the maps

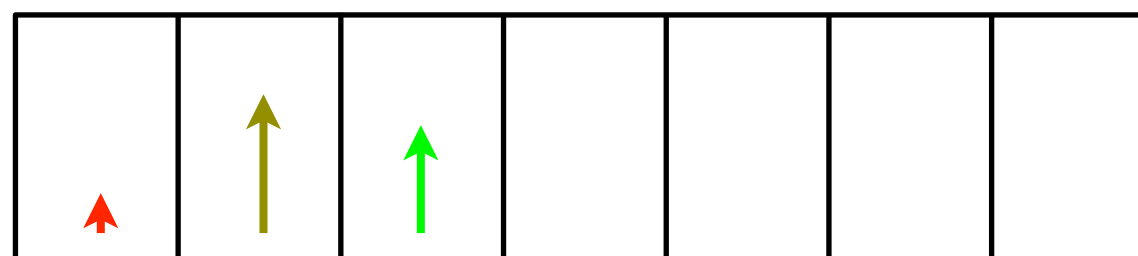
Danger



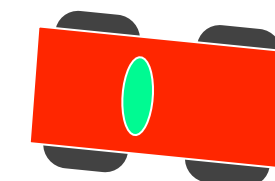
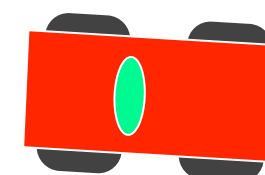
X X X X



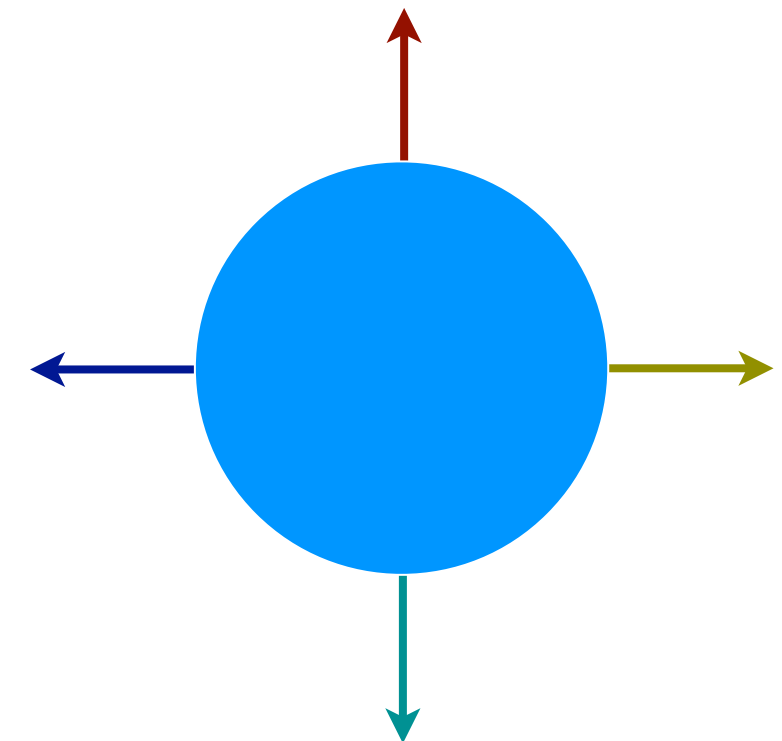
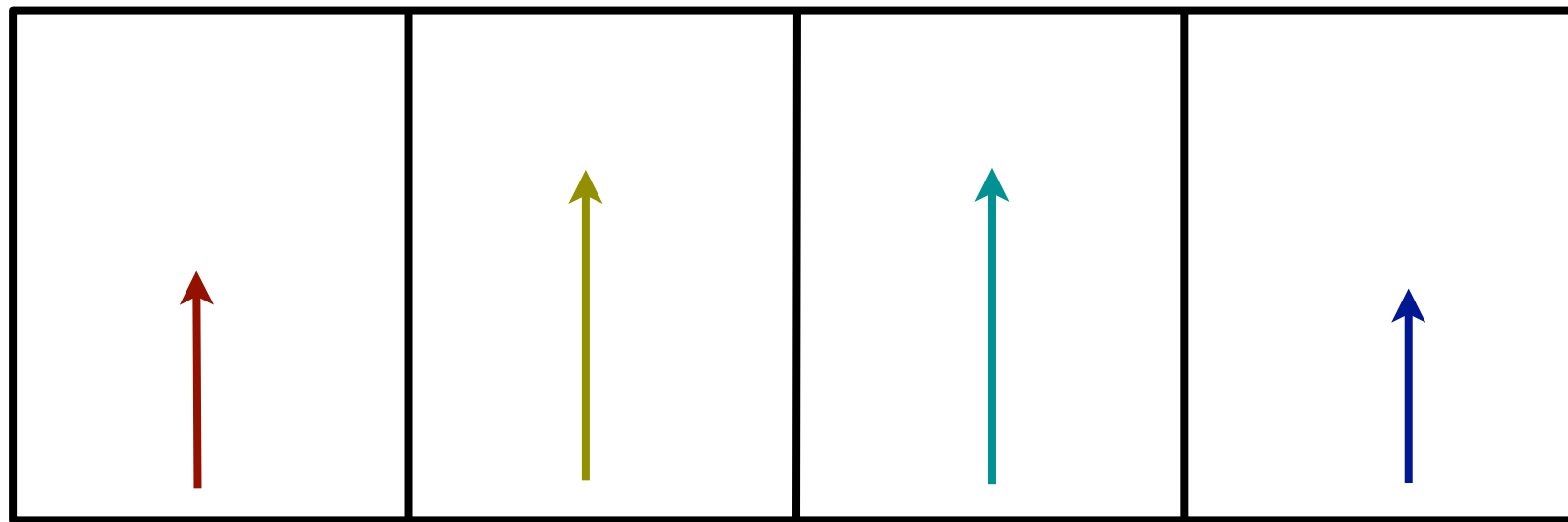
Interest



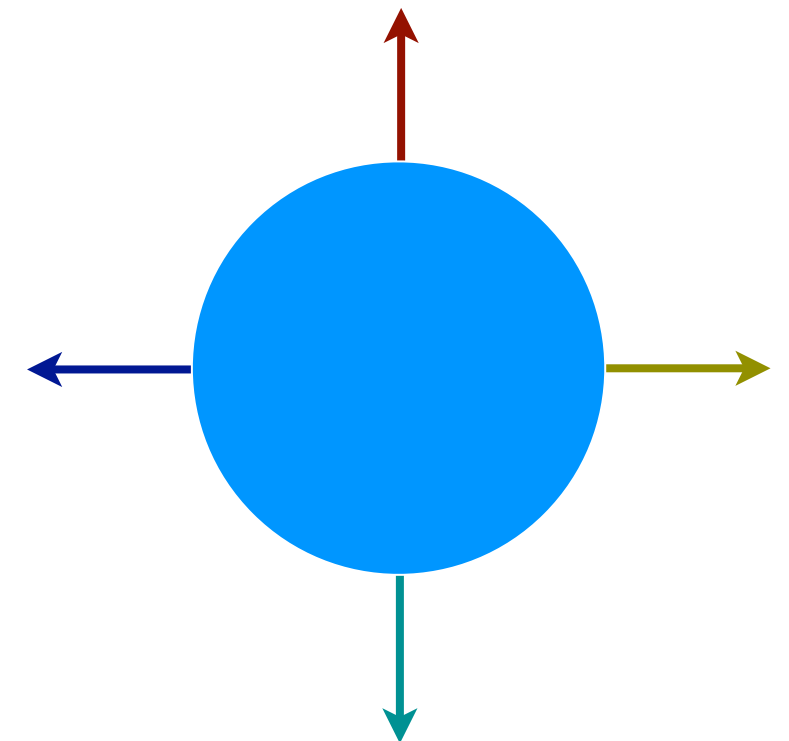
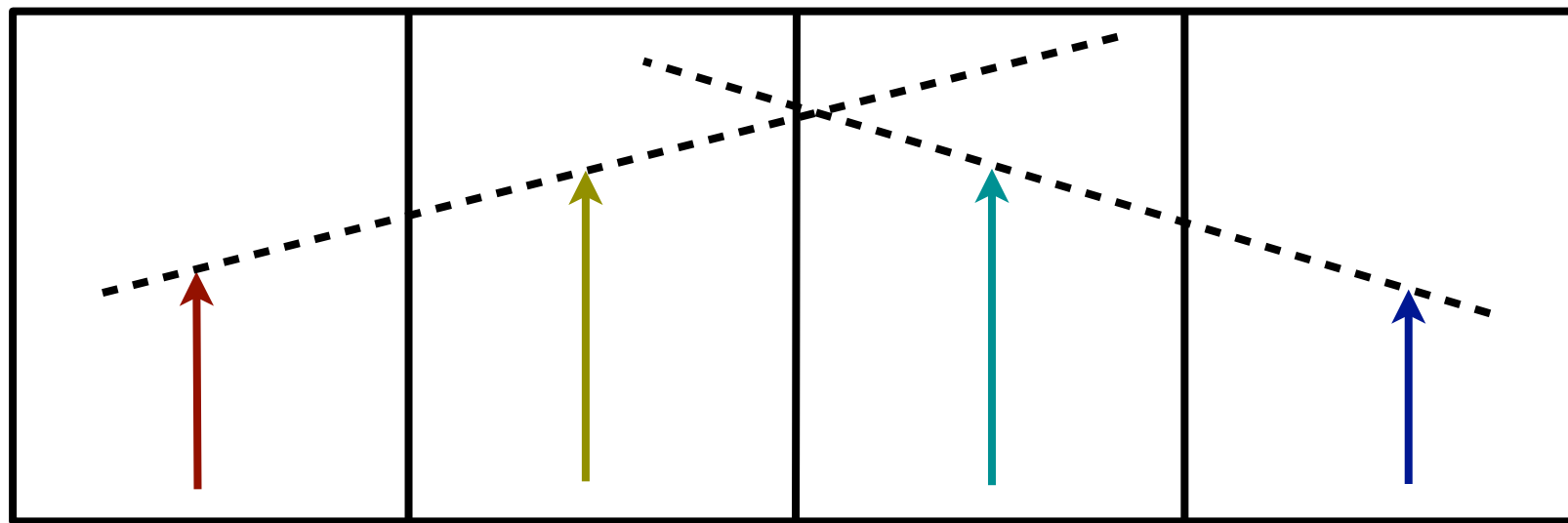
X X X X



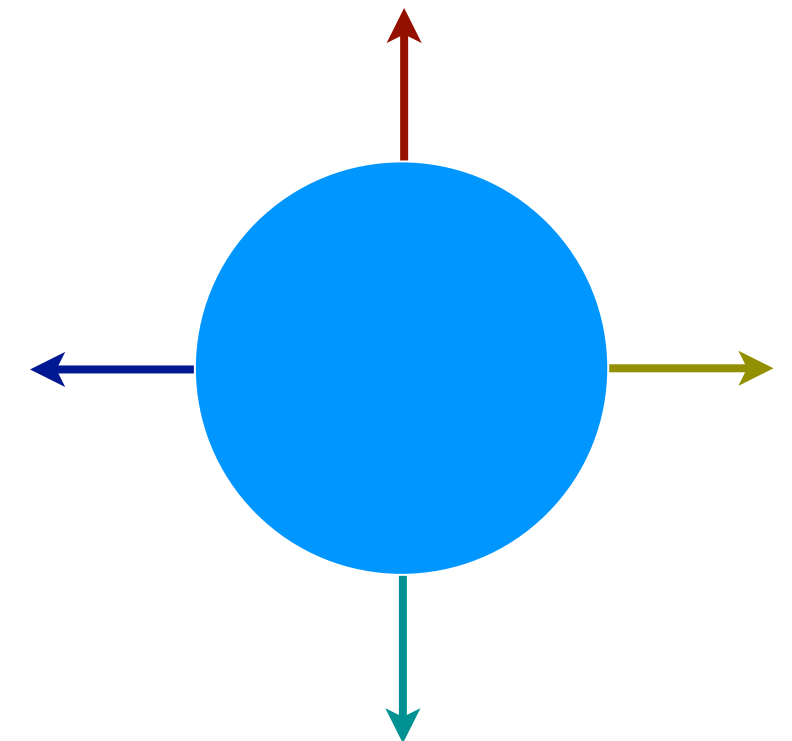
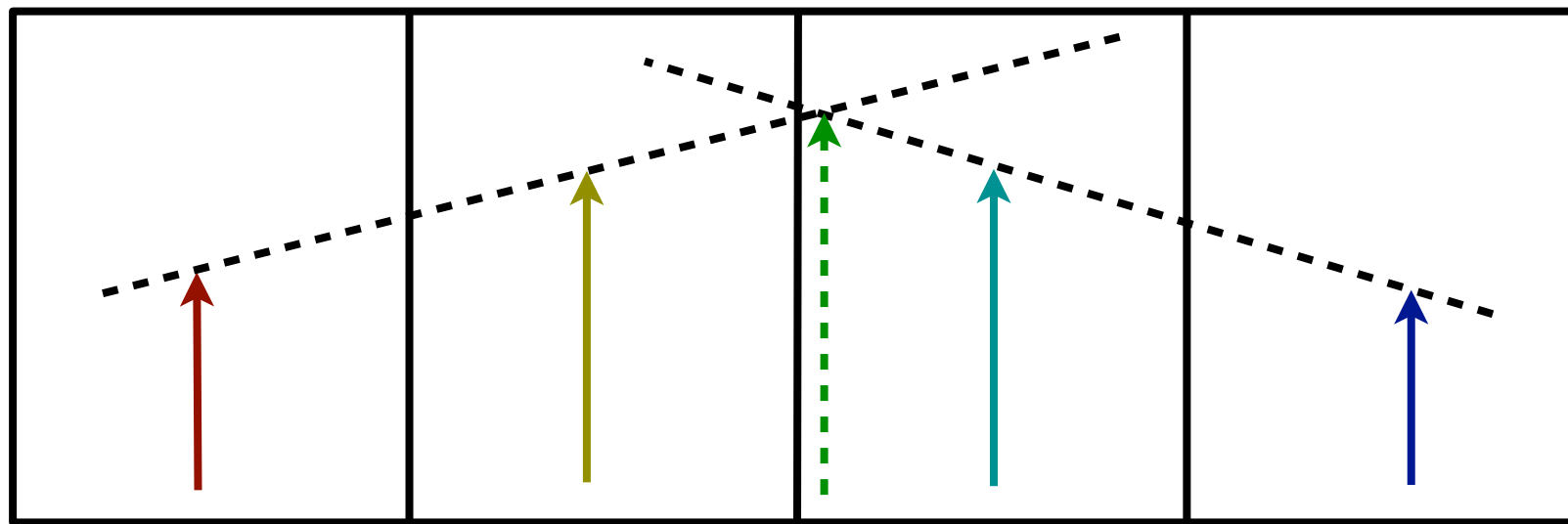
Smoothing decisions



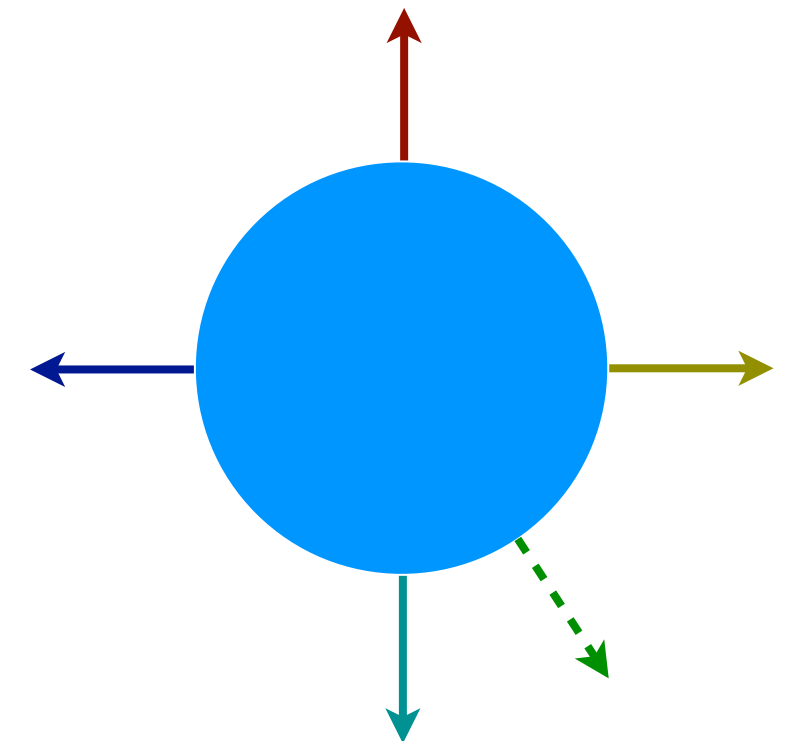
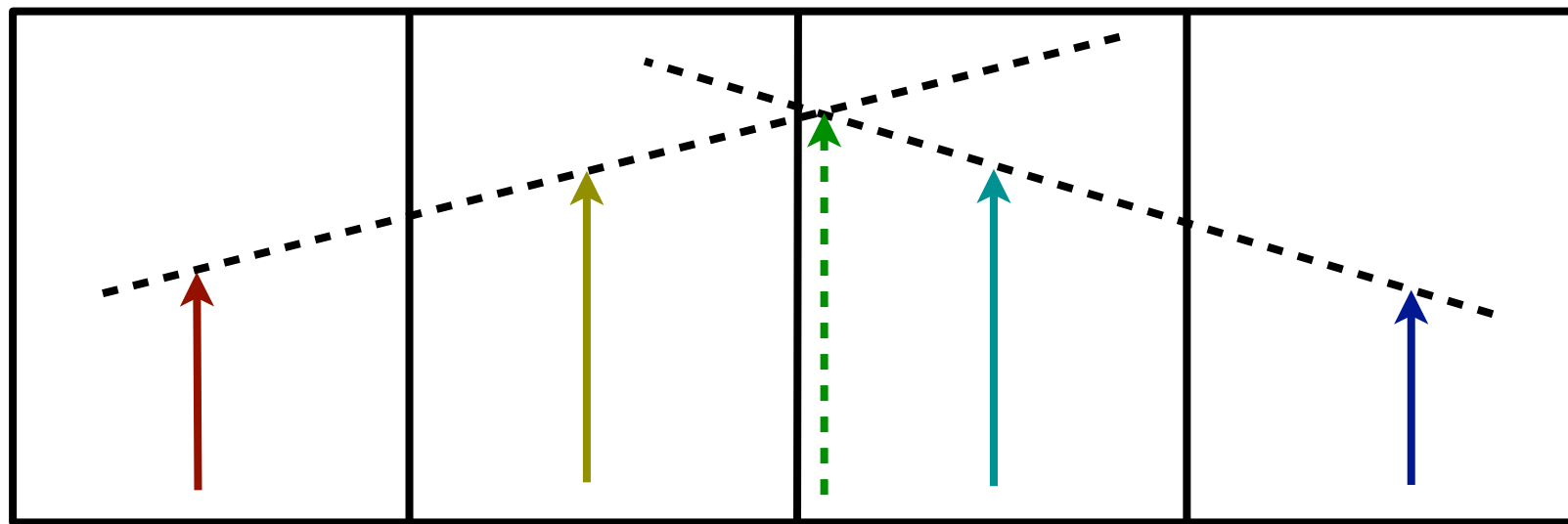
Smoothing decisions



Smoothing decisions



Smoothing decisions



Performance



Performance

- Linear to context map size, behaviours



Performance

- Linear to context map size, behaviours
- LOD out low priority behaviours

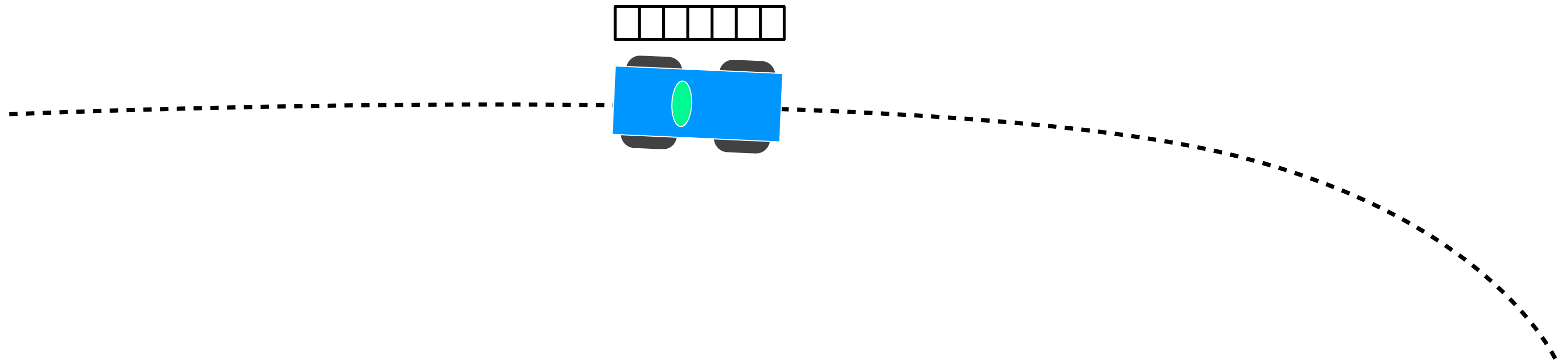


Performance

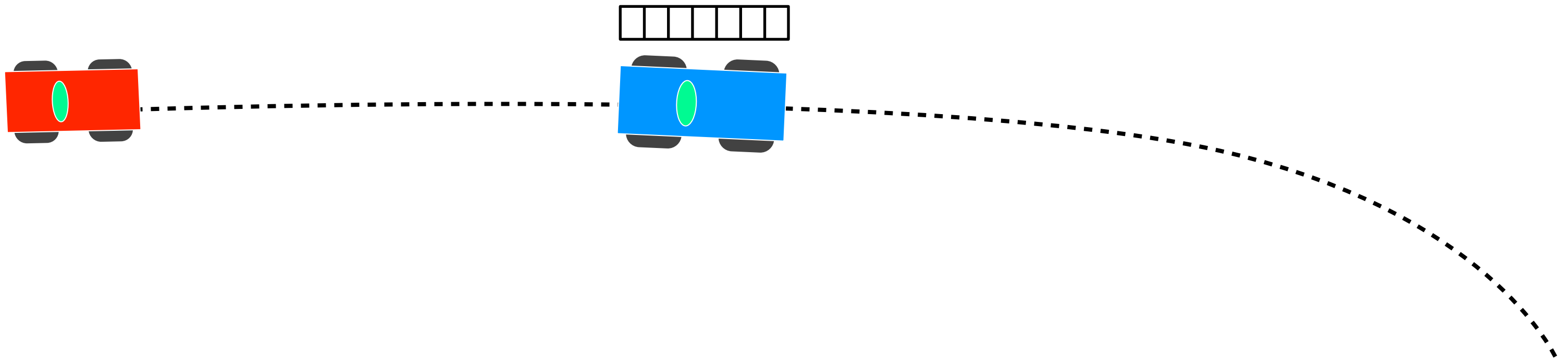
- Linear to context map size, behaviours
- LOD out low priority behaviours
- Vectorisation



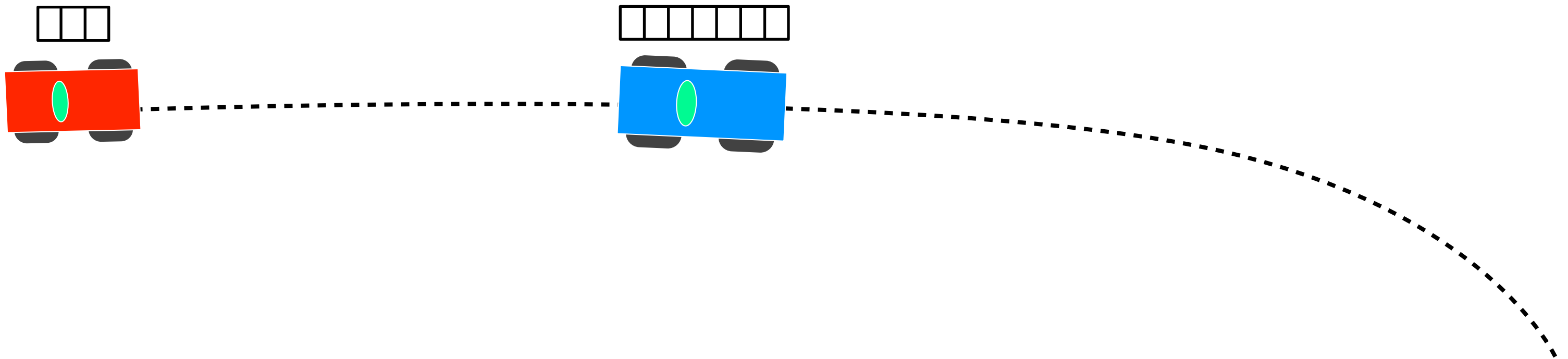
LODing



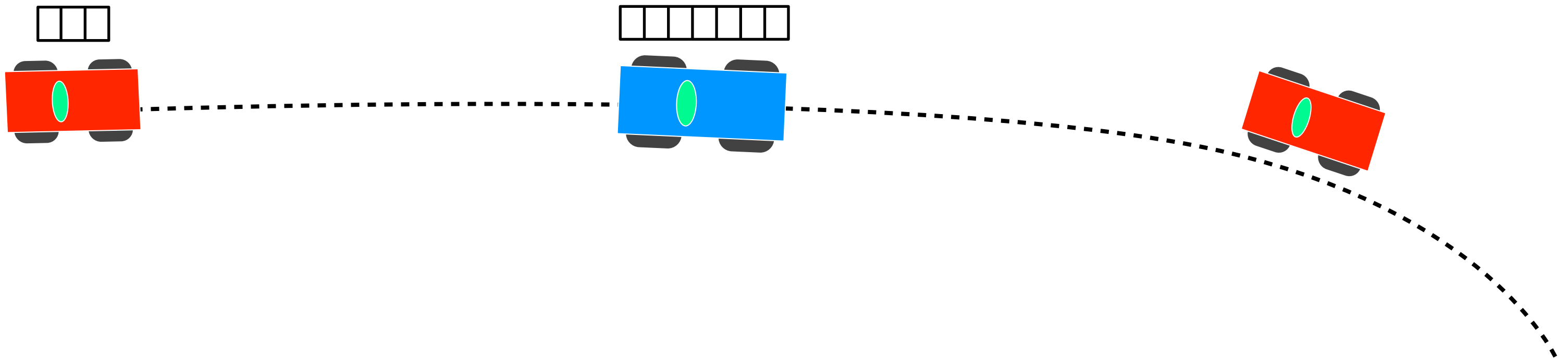
LODing



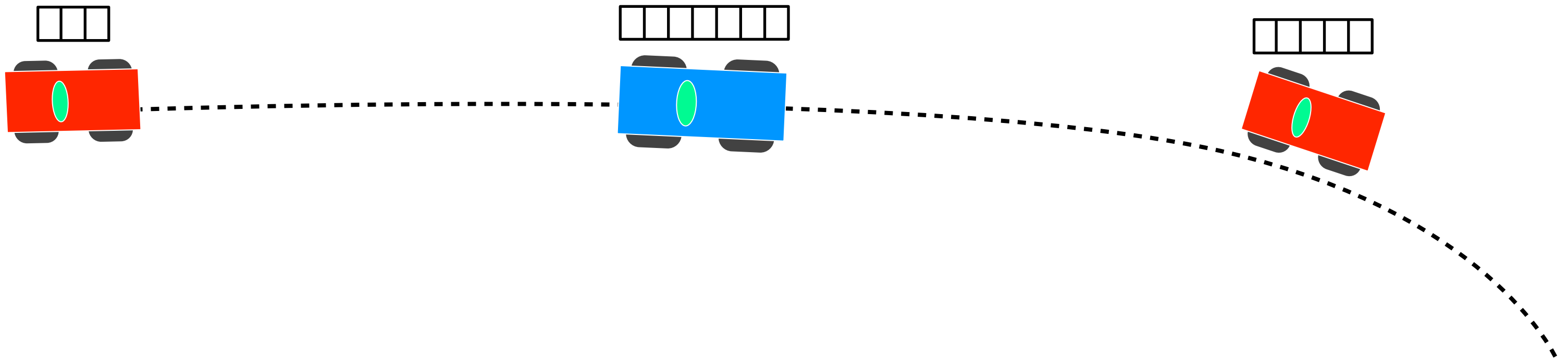
LODing



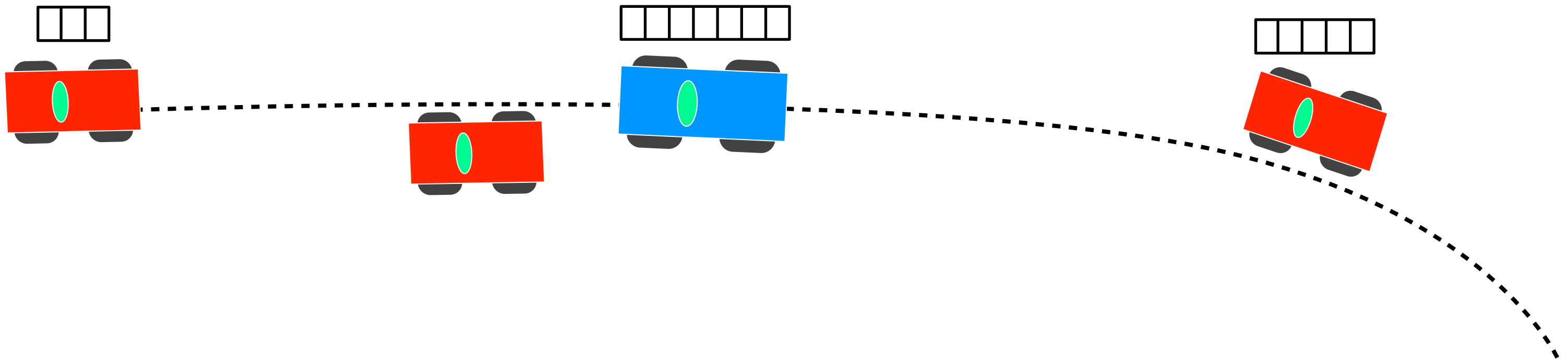
LODing



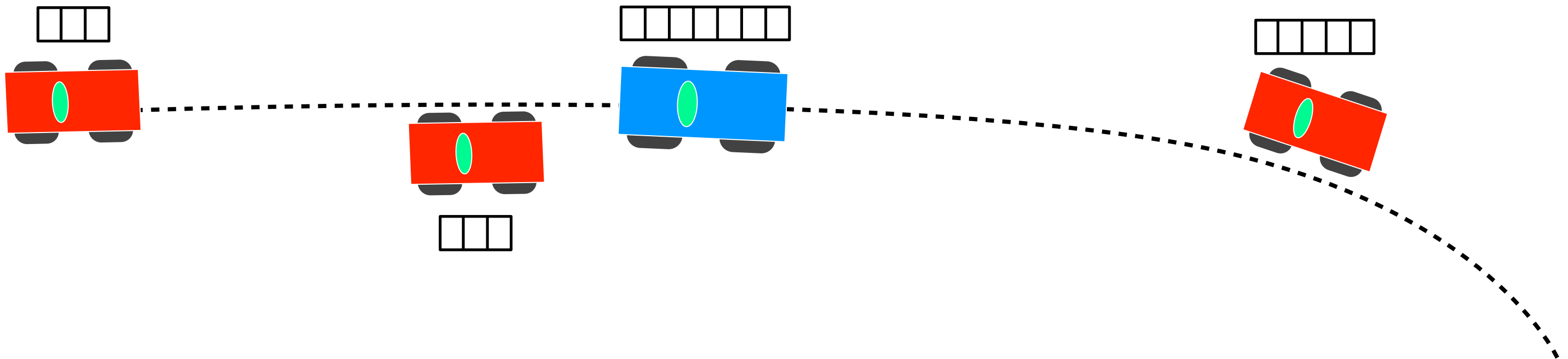
LODing



LODing



LODing



Pre-processing Context Maps

Pre-processing Context Maps

- Smoothing

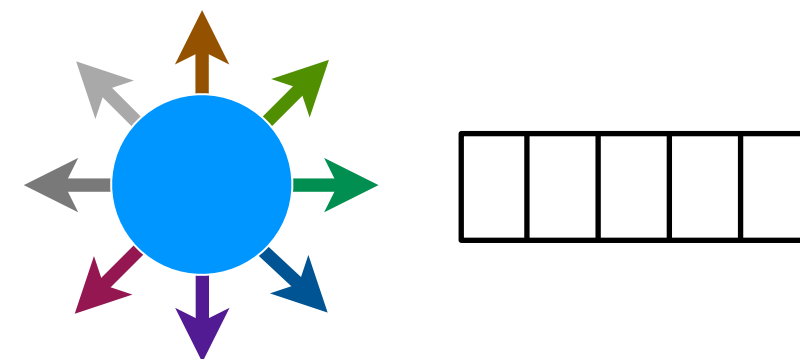
Pre-processing Context Maps

- Smoothing
- Blend with previous frame

Recap

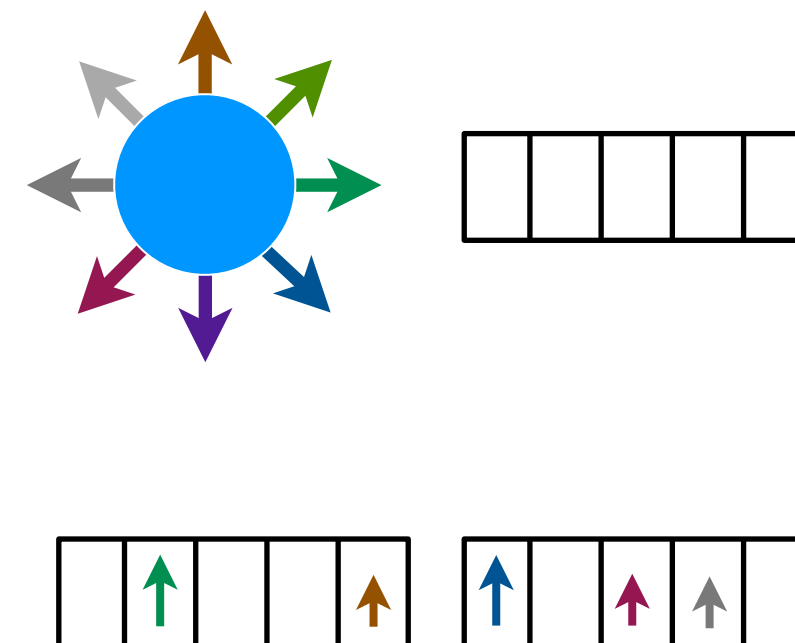
Recap

- Create projection from decision space to 1D context map



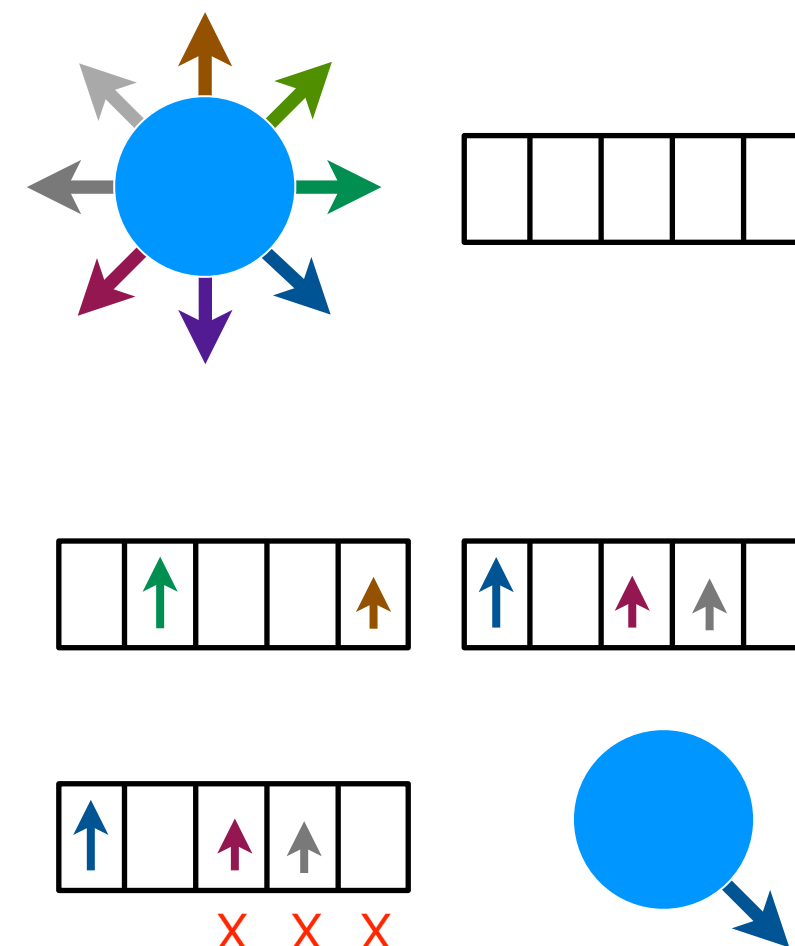
Recap

- Create projection from decision space to 1D context map
- Behaviours write world view into danger, interest maps



Recap

- Create projection from decision space to 1D context map
- Behaviours write world view into danger, interest maps
- Evaluate maps for best decision



Conclusion

Conclusion

- Small, stateless, decoupled behaviours

Conclusion

- Small, stateless, decoupled behaviours
- Separated WHAT from HOW

Conclusion

- Small, stateless, decoupled behaviours
- Separated WHAT from HOW
- Emergent behaviour

Conclusion

- Small, stateless, decoupled behaviours
- Separated WHAT from HOW
- Emergent behaviour
- Guaranteed movement constraint

Conclusion

- Small, stateless, decoupled behaviours
- Separated WHAT from HOW
- Emergent behaviour
- Guaranteed movement constraint
- Consistent decisions

Conclusion

- Small, stateless, decoupled behaviours
- Separated WHAT from HOW
- Emergent behaviour
- Guaranteed movement constraint
- Consistent decisions
- Suitable for macro scale

Andrew Fray

- @tenpn
- andrewfray.wordpress.com
- about.me/andrew.fray

Graham Pentheny

- @grahamboree
- graham.pentheny@gmail.com
- slides available at:
grahampentheny.com/gdc