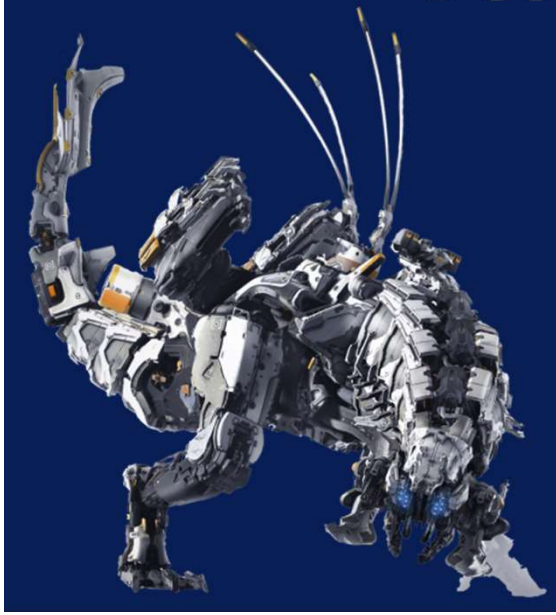
PON9
Playlab fOr inNovation in Games

UNIVERSITÀ DEGLI STUDI DI MILANO
DIPARTIMENTO DI INFORMATICA

AIVG:
Designing games AI

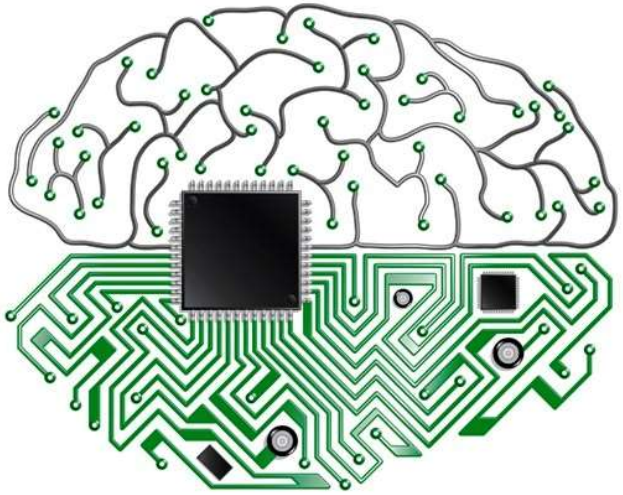


Laura Anna Ripamonti

Summary

- **Goal**: understanding how different techniques and algorithms in the field of Artificial intelligence can be profitably exploited to obtain specific gameplays

1. Designing AI for a video game
2. AI & video games:
   - AI & Shooters
   - AI & driving games
   - AI & RTS
   - AI & turn-based strategy games
   - AI & sport games



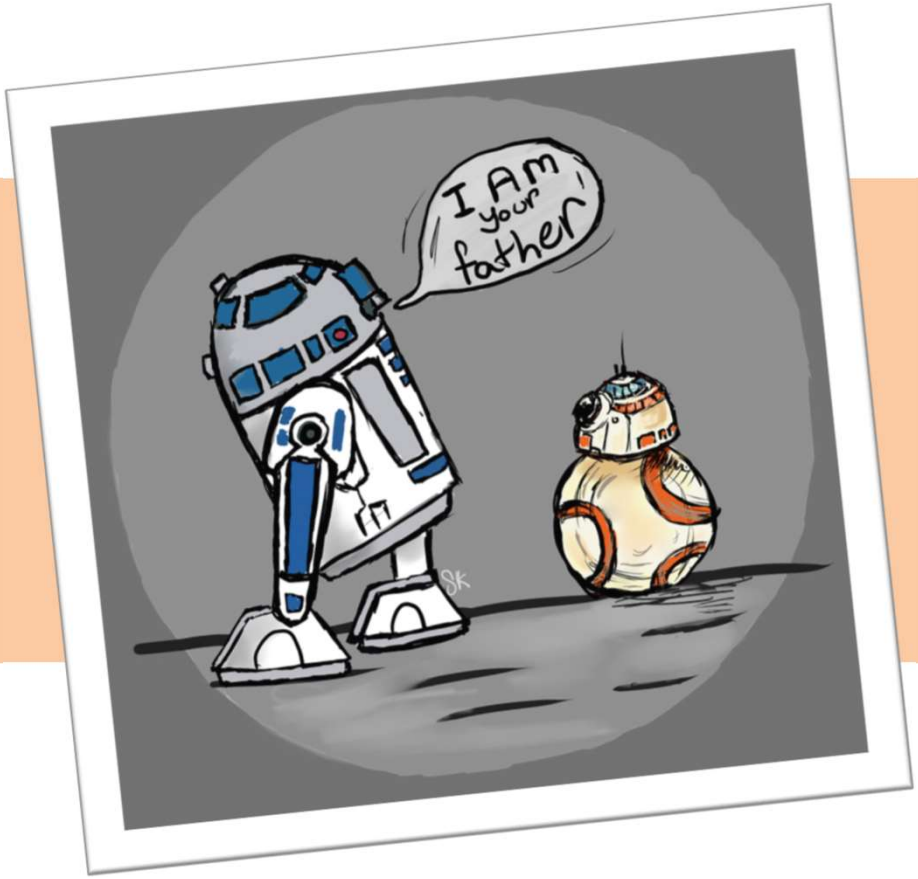
- References: I. Millington, J. Funge *Artificial Intelligence for Games*. Morgan Kaufmann

Designing games AI

Designing AI for a video game

AI in games … BELIEVABILITY

- … is all about convincing players that they are dealing with a «real intelligence»!



BUT … generally it is **NOT** about optimization …

AI goals in games

AI in games fulfills two main purposes:

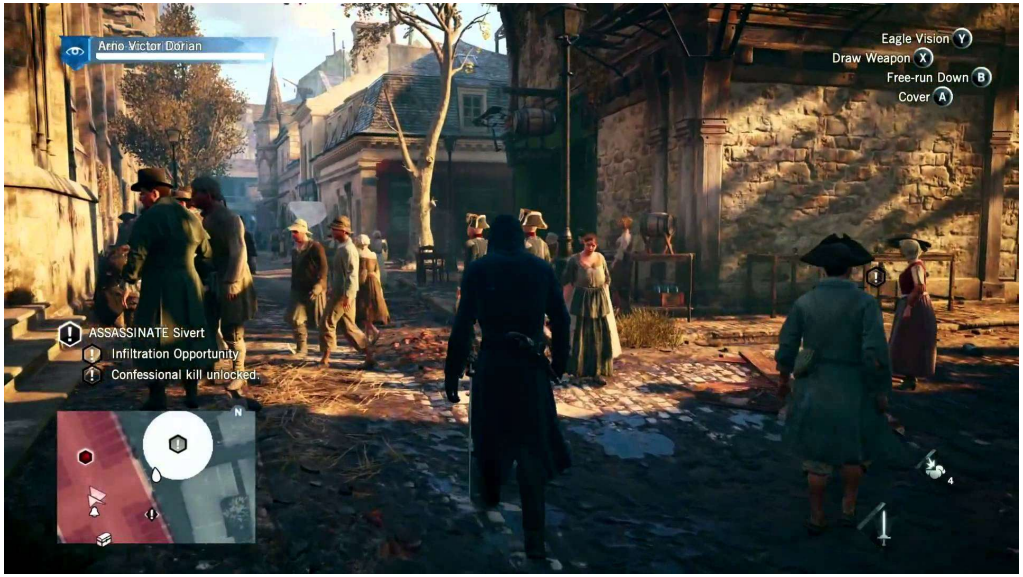
1. **Assisting** gameplay

2. **Enhancing** immersion
   - Include psychology of the agent(s)

AI & Game Design

- There are **many types and uses of AI**, several may become **CRUCIAL for the success of the game**

- When using agents/AI, the Game Designer defines:
  - the **psychology** of the creature(s)
  - Where to **place** them
  - How to make their **behaviour readable/ scary/etc**.
  - How to **avoid predictabilty**
  - Etc.



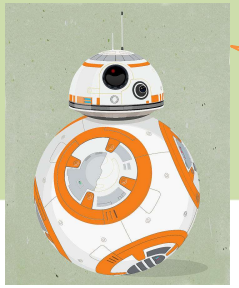
Designing AI for video games (ideally …)

PUBLISHER MILESTONES !!!!

DESIGNING AI PROCESS

1. Work out **desired behaviours** from the Game Design Doc.
2. Select the **simplest set of techniques** supporting them
    1. Plan how to integrate them
    2. Plan how to integrate AI and the game engine
    3. Put placeholders for character behaviours
3. Start to work ….

QUICK & DIRTY CODE PACKED IN THE FINAL VERSION



1. Evaluating characters behaviour

- Description made by the game designer

- Behaviours are NOT set in stone, they evolve during the game development (a game is a dynamic system…)
  - Develop with flexibility in mind
  - Optimize code only at the end

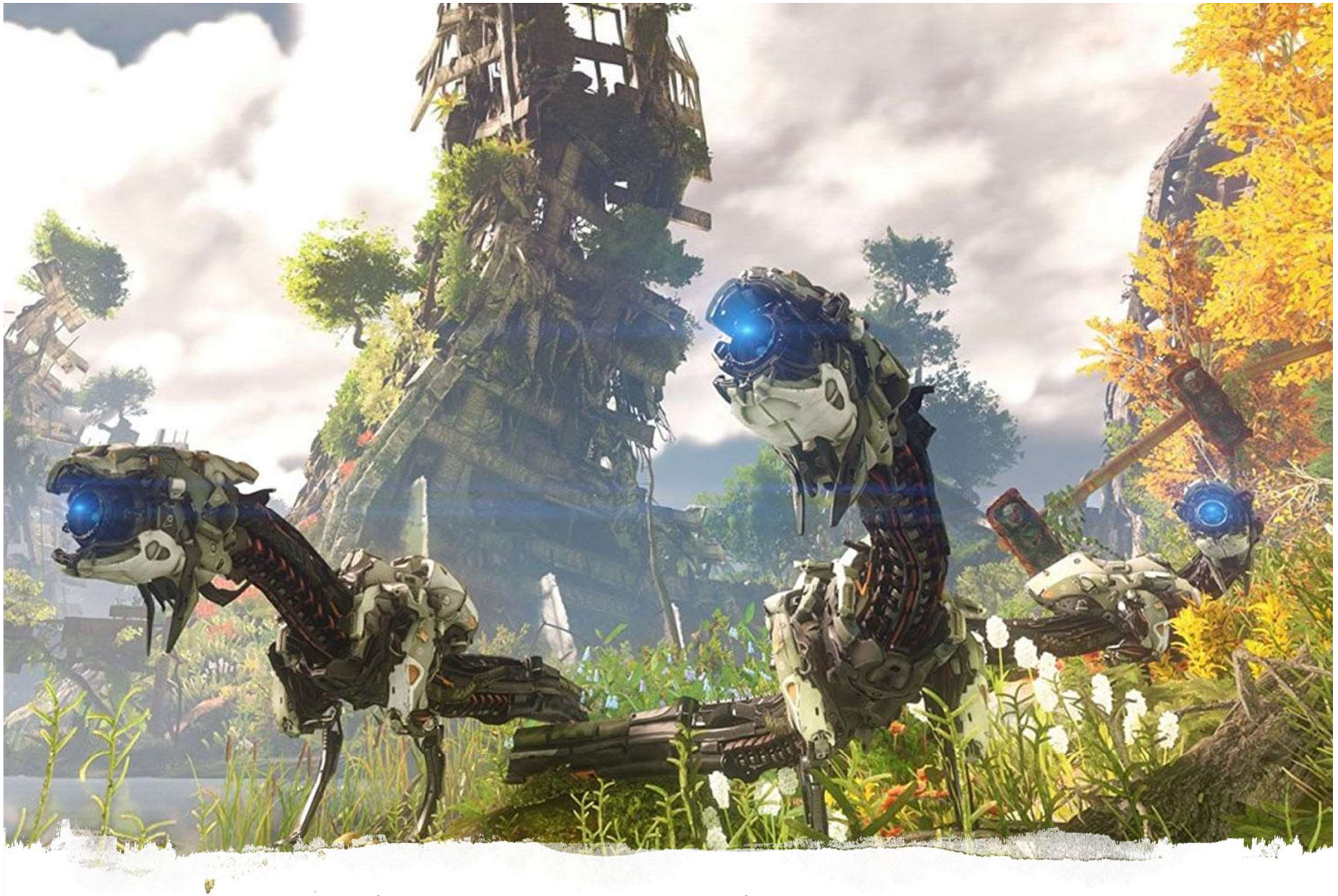
1. Evaluating characters behaviour

3 main groups of questions to answer:

a. Movement

- Individuals or groups?
- Degree of realism?
- Any (realistic) physical simulation needed?
- Any pathfinding needed?
- Characters' motion influenced by other characters?

This monster (Horizon zero dawn) can be tamed => its behaviour changes totally

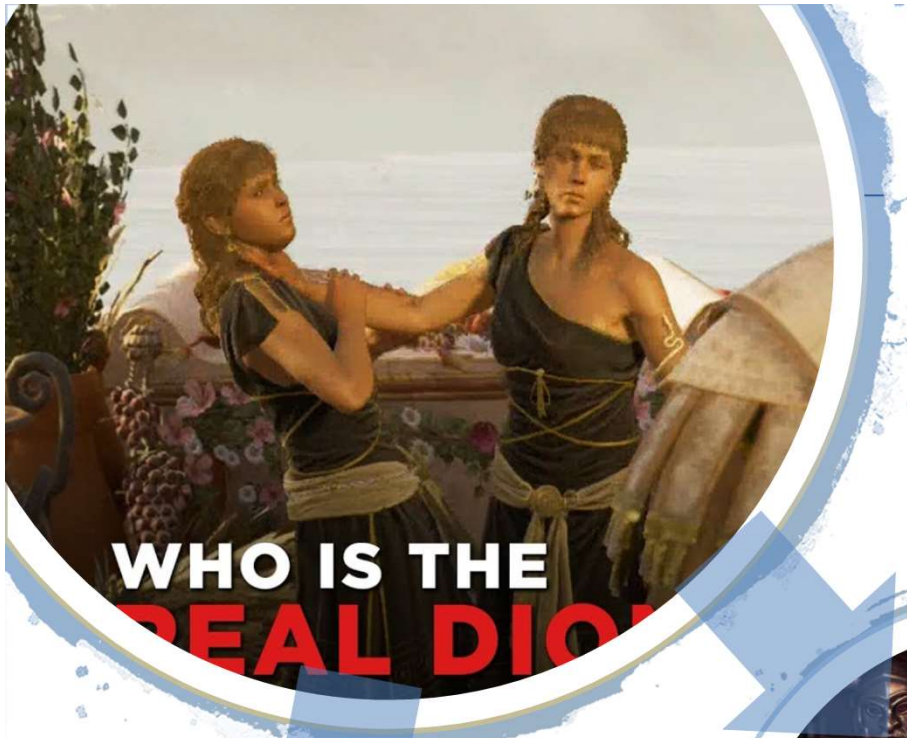


This type of monster (from Horizon zero dawn) moves in flocks
=> they need some type of coordination

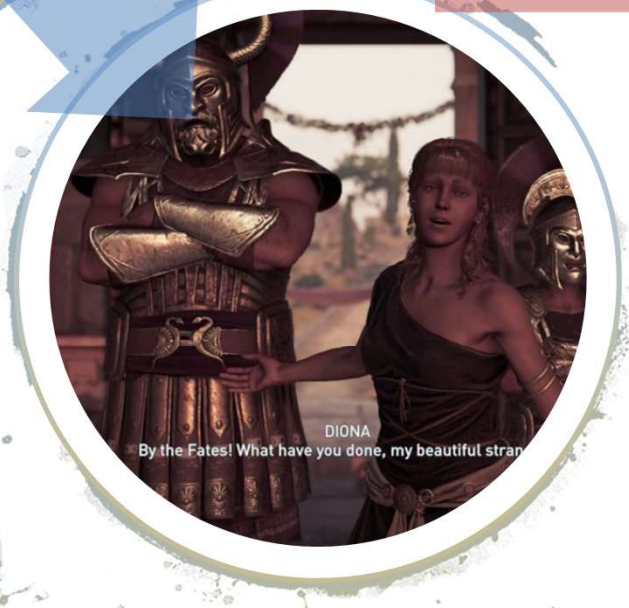
1. Evaluating characters behaviour

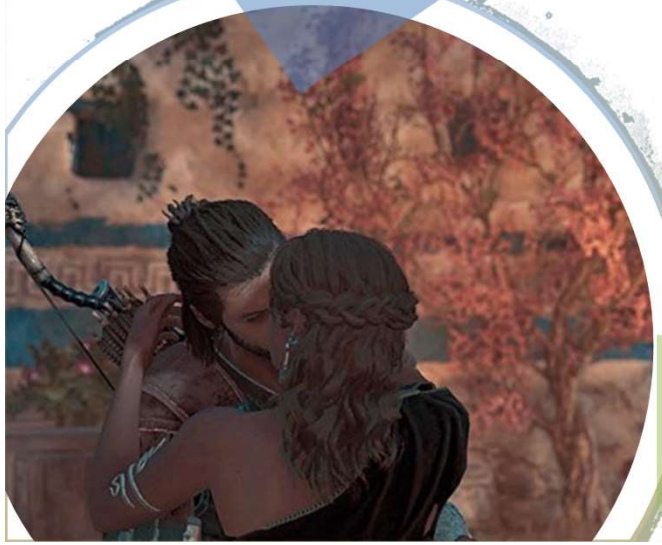
b. Decision making

- Full range of different possible actions?
- How many distinct states for each character? (how actions will be grouped togheter?)
- When and why will the character change its behaviour?
- Characters need to look ahead to make decision
- Character decisions are influenced by the players actions?



U killed the wrong one





U killed the right one

Diona dilemma - Assassin's Creed Odissey

Agents' behaviour changes as a consequence of your decisions

1. Evaluating characters behaviour

- Questions to answer:

c. Tactical and strategic AI

  - Characters need to understand the game on a large-scale?
  - Characters work together?
  - Characters think for themselves and still display a certain pre-defined group behaviour or not?
  - Do you need decisions for specific groups of characters?

Starcraft: agents need to cooperate on a large, strategical scale!

2. Selecting techniques

- Once the behaviours are set, we have to select candidate **techniques** to implement them

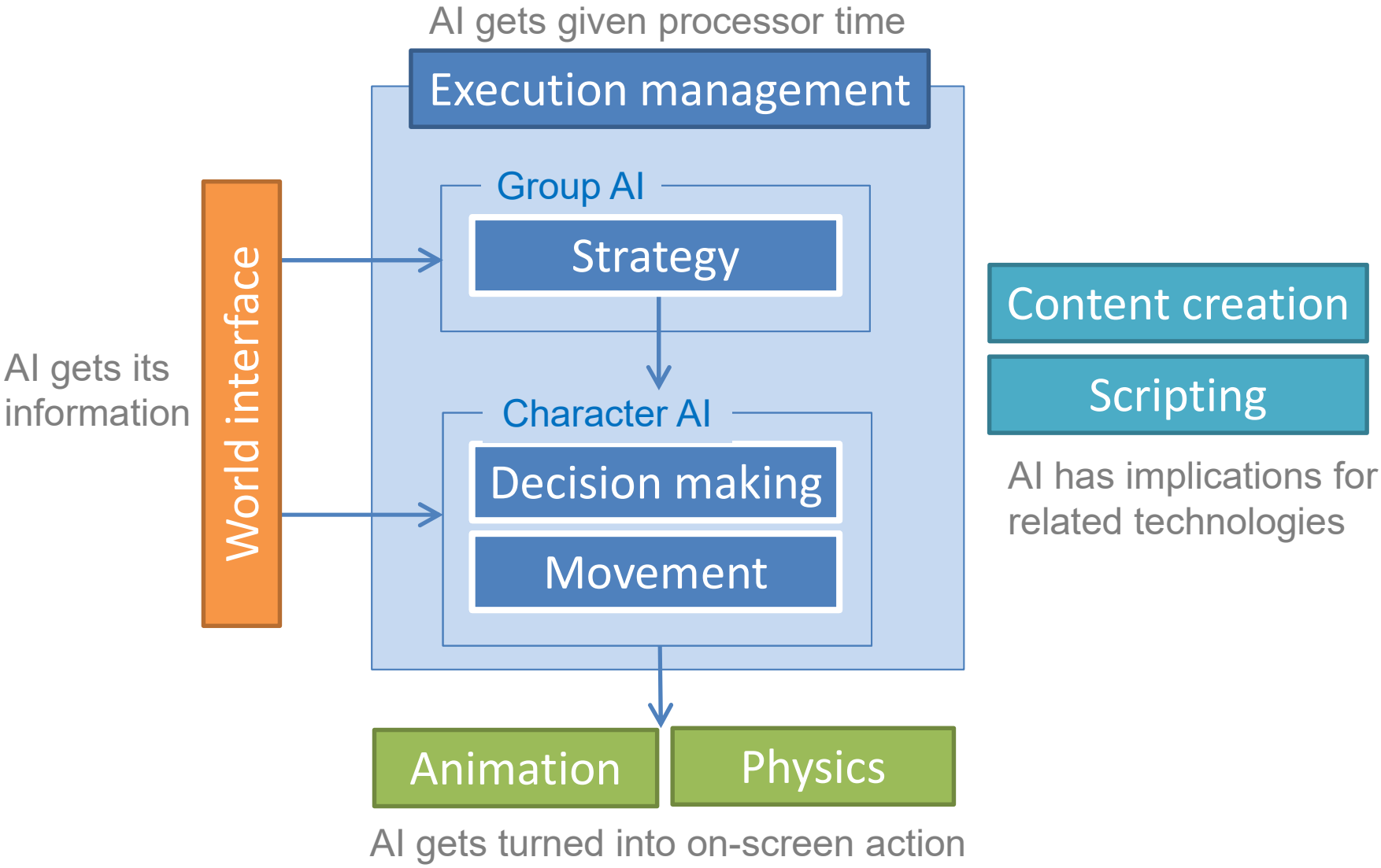
- Critical point: decision making system
  - Try first with the simplest approaches (behaviour trees, state machines)
  - Avoid exotic, complex techniques unless REALLY needed

2. Selecting techniques

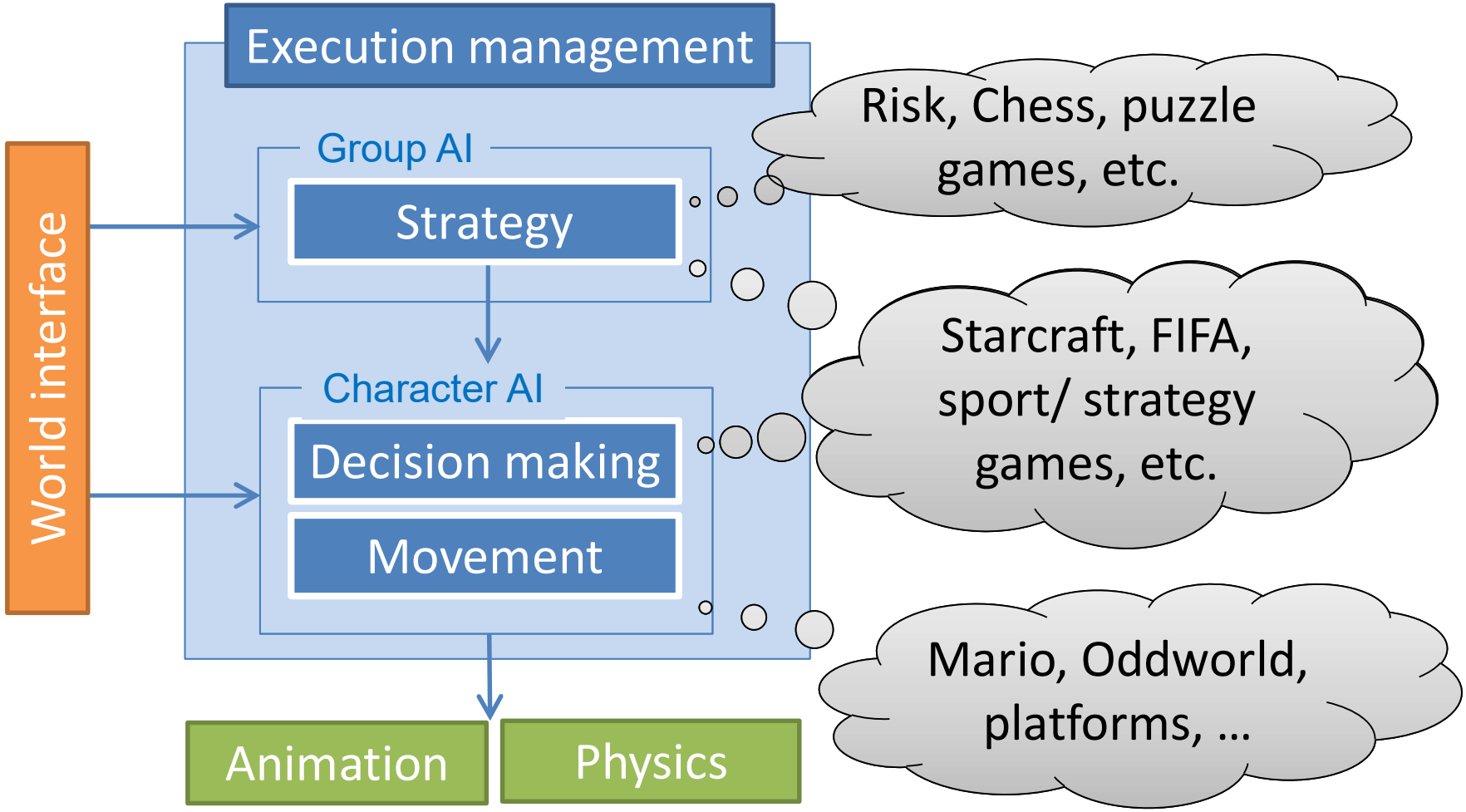
- It exists a large zoo of AI techniques …
  => better avoiding to get lost

- Try to understand **how the bits fit together** by trying to fit each technique into a general structure for making intelligent game carachters …!

2. Selecting techniques: AI model for video games



2. Selecting techniques: AI model for video games

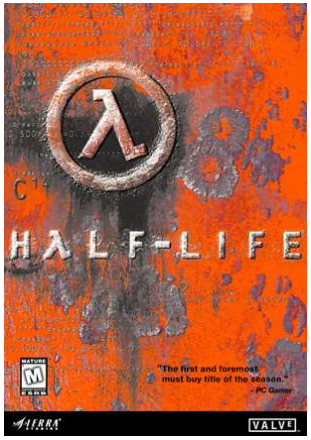


2. AI model for video games: strategy

- «**Strategy**»: coordinates a whole team

- Each character has its own decision & movement algorithms, but their decision making will be influenced and guided by a group strategy

- (Half-Life and Tom Clancy's Ghost Recon – among the first examples of enemies which coordinates in order to surrond effectively the player)



2. AI model for video games: decision making

- «**Decision making**»: a charachter works out what to do next

- Decision making algorithms can be:
  - Very simple (an animal moves away only when the player gets very near)
  - Very complex (intermediate actions can be chained together to achieve a goal, such as laying down bombs, etc.)
- Each character has a range of possible behaviours
- Once the behaviour has been chosen, it can be executed by animation & movement techniques (e.g. a melee attach) or even have no visual feedback (e.g. Civilization)

2. AI model for video games: Movement

- «**Movement**»: algorithms that turn decisions into some kind of motion

- Movement algorithms can be:
  - very simple (homing in & c.)
  - very complex (sentinel detecting the player raises an alarm in Splinter Cell, obstacle avoidance, etc.)

- NOTE: a lot of actions are carried out using ANIMATION DIRECTLY
  (if a Sim sitting at the table with food in front of it desires to eat, the eating animation is played. Once the decision to eat has been taken, no more AI is needed!)

2. AI model for video games: infrastructure

- Note that to build AI for a game, algorithms alone are **NOT** enough:

- Movement requires animation and physics simulation
- AI needs information about the game world (perception)
- AI must be managed to optimize CPU & memory usage (this imply a totally different set of algorithms will be brought in play….!)

2. AI for video games …

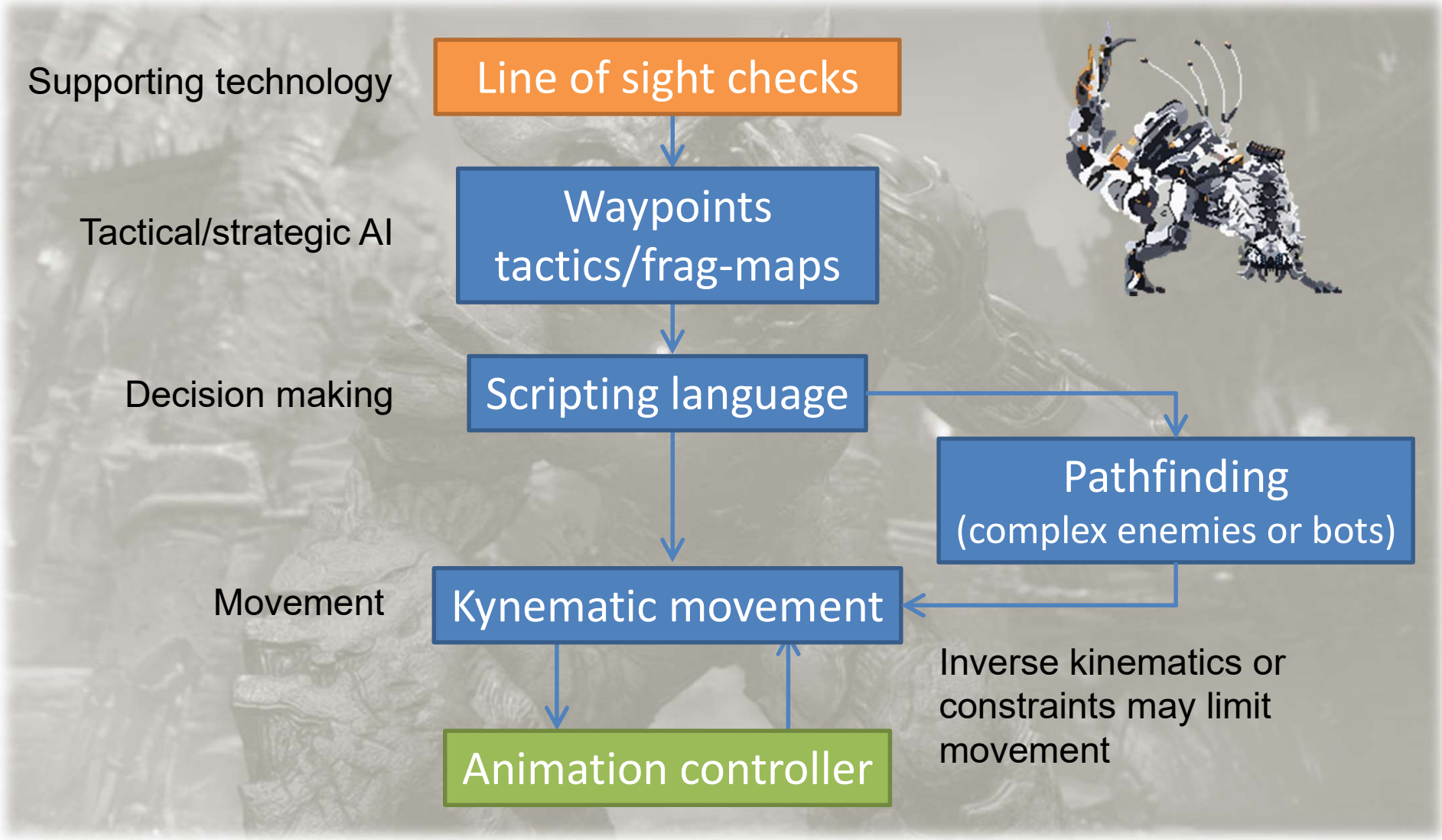
Designing games AI

AI and Shooter games

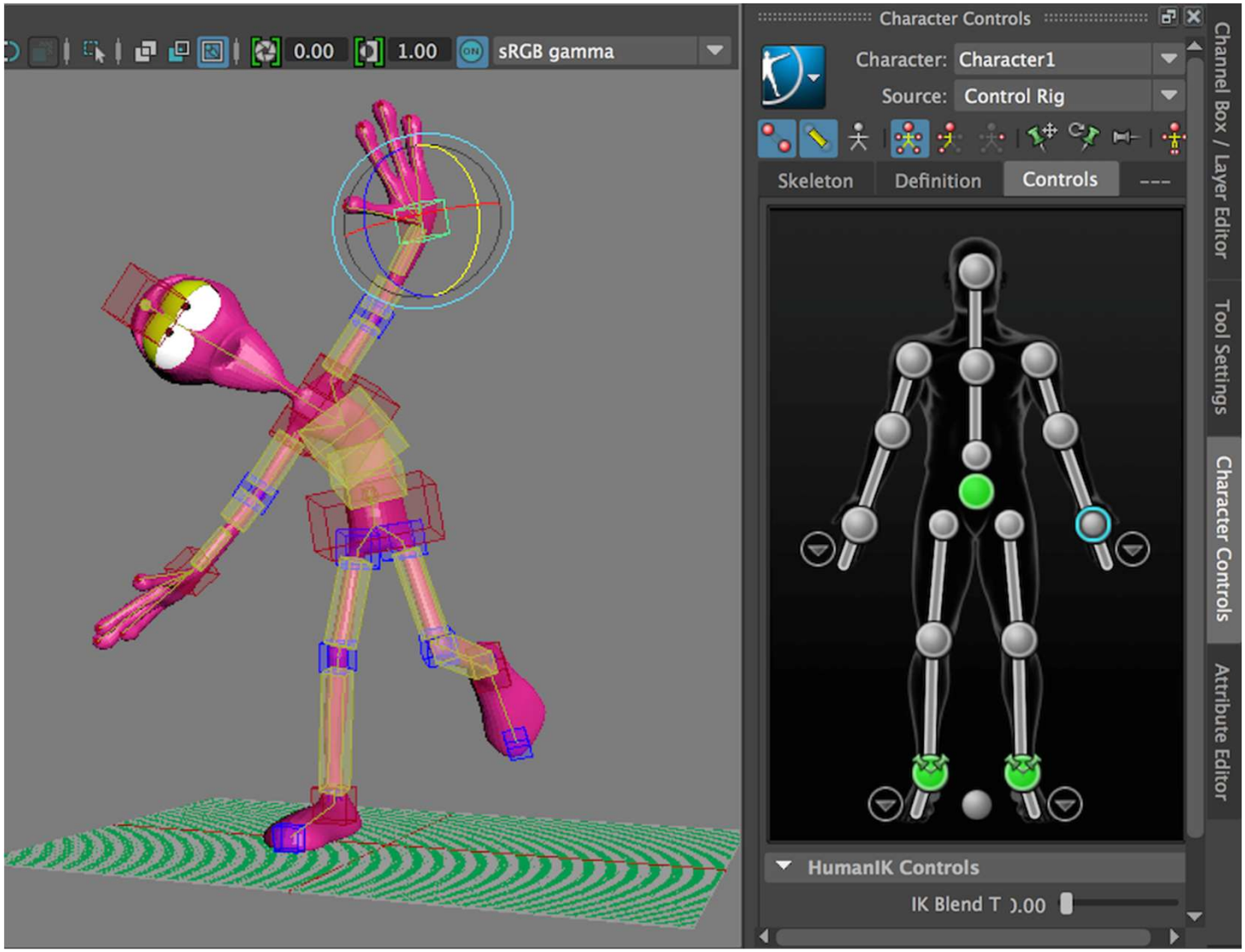
Main AI needs for Shooter games

1. Movement
   - Control enemies
2. Firing
   - Accurate fire control
3. Decision making
   - Generally state machines
4. Perception
   - Who to shoot and where they are
5. Pathfinding
   - NPCs plan they route
6. Tactical AI
   - NPCs find safe positions or ambush

AI architecture for a Shooter



Inverse kinematics ….

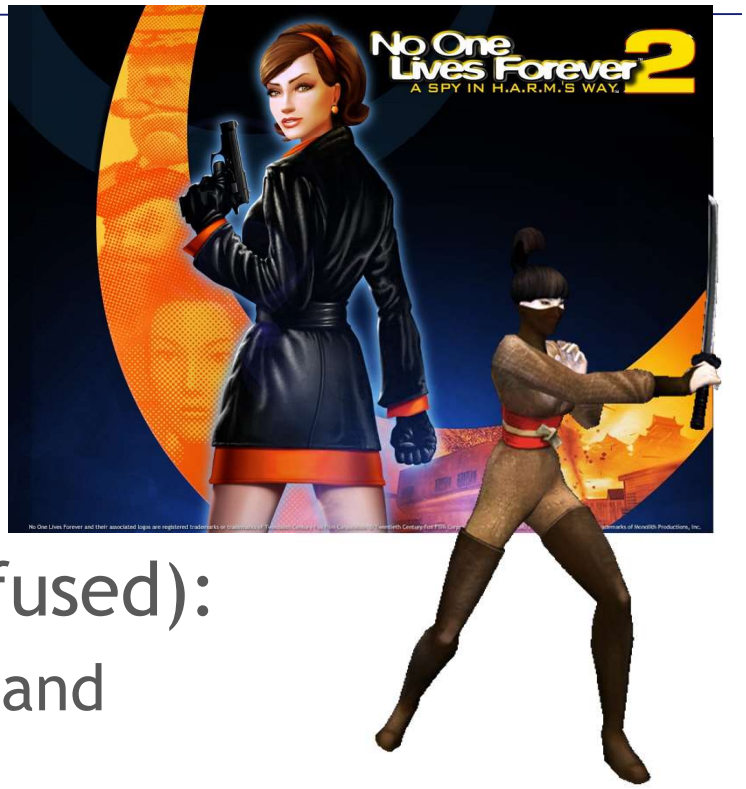


Shooters: Movement and Firing

- Movement:
  - **Most visible part of character behaviour**
  - Shooters have the most complex sets of animations (tens of hundreds of sequences combined, plus inverse kinematics and ragdoll physics)

Shooters: Movement

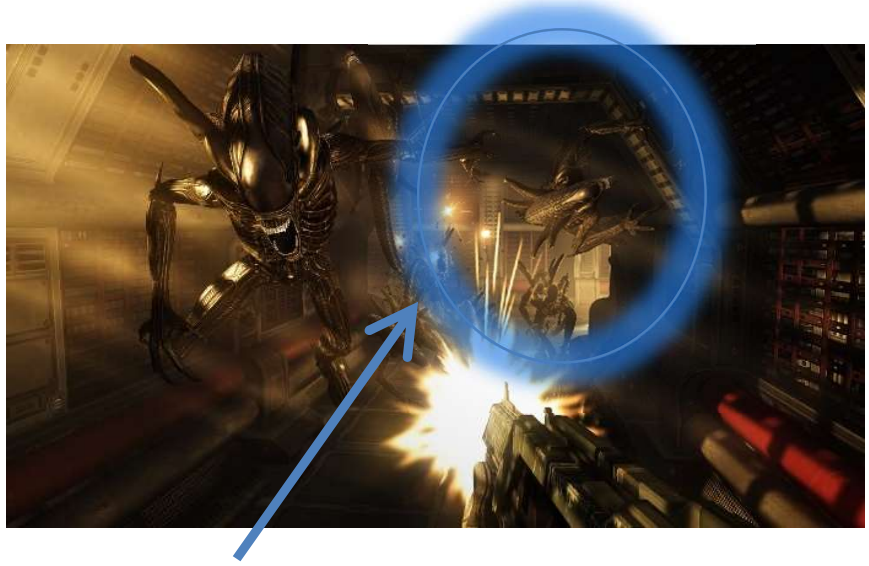
- Moving around the level **IS** a challenge, AI needs to:
  - Work out the route
  - Break motion into animations



- Possible approaches (more diffused):
  1. **AI split in 2 parts**: pathfinding and animation management
  2. Games using scripting languages use the **same controls as the players**: AI needs to specify only direction, velocity, weapon changes, etc.

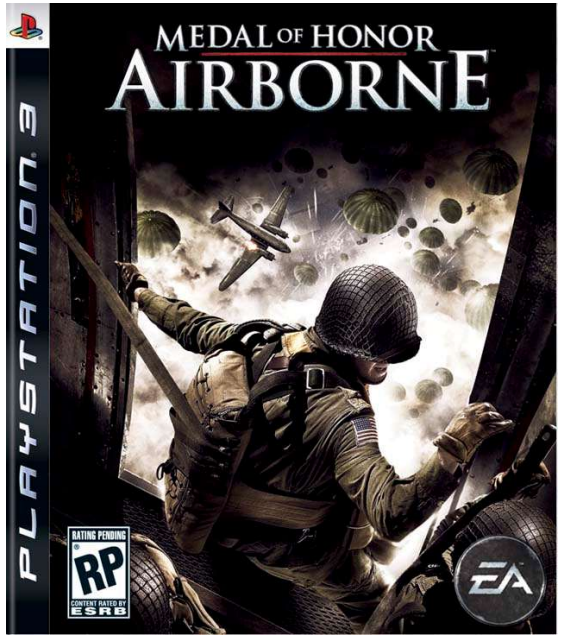
Shooters: Movement

- Indoor dynamic settings:
  - characters have to find a way in **constrained settings** while **reacting** to other characters/player moves: es. **pathfinding + repulsion** force between all characters similar to collision systems
- Open areas (e.g. space):
  - **Pathfinding** + **collision system** + **formation motion system**
- NB: the «floor» can be the ceiling as well … (Halo The Flood, Alien vs Predator, …)



Shooters: Firing

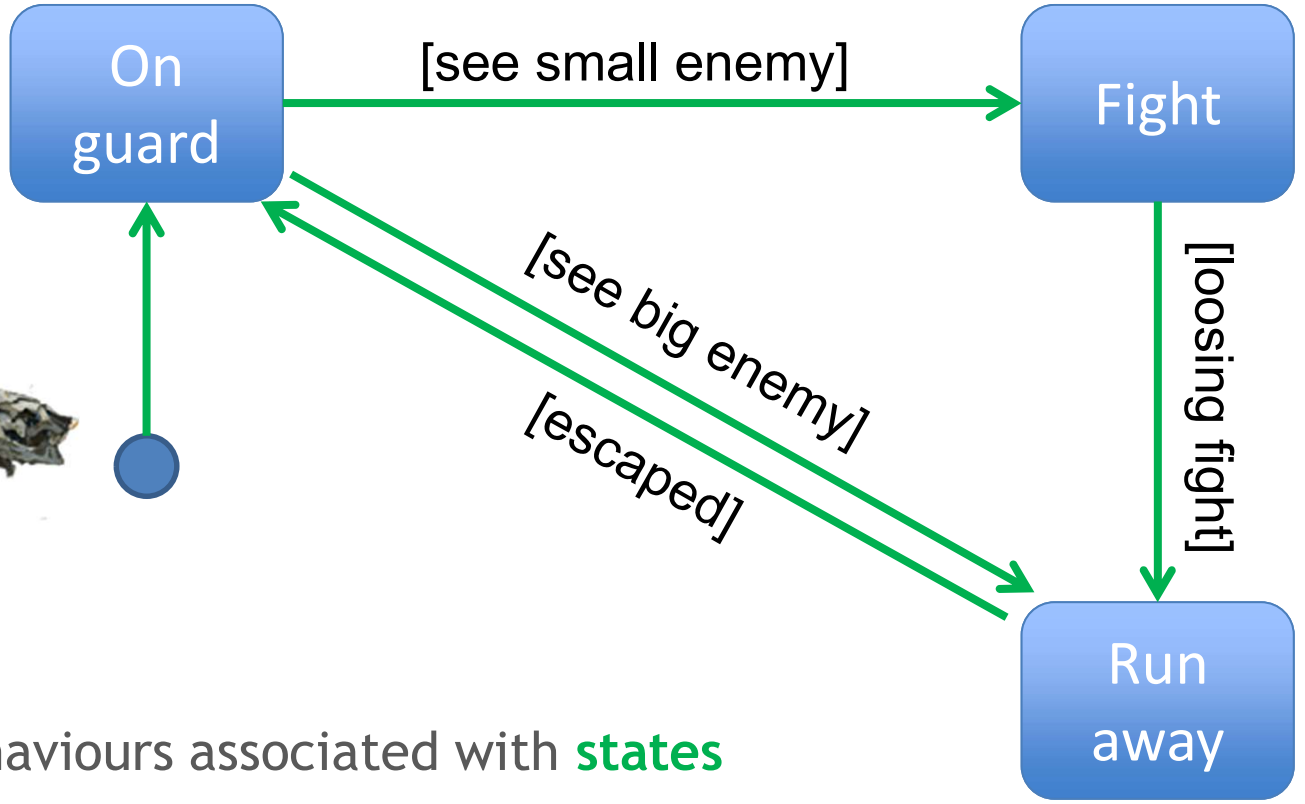
- It's important to guarantee a good trade-off between **accuracy** and **fun**:
  - *DOOM* (id software, 1993) criticized for excessively accurate aiming (developers had to slow down projectiles to assure players a possibility to survive)
  - *Medal of Honour: Airborne* (EA 2007) uses firing models allowing characters to miss in exciting ways



Shooters: decision making

- Commonly achieved through:
  - **Finite state machines:** not every action is reachable, only the ones connected to the current state
  - **Behaviour trees:** any decision can be reached through the tree
- Common approach: bot scripting system
  - Script (in game-specific language) that polls the current game state and asks actions to be executed (animations, pathfinding, movement)
  - Scripting language may be available to players (Unreal)

FMSs: example



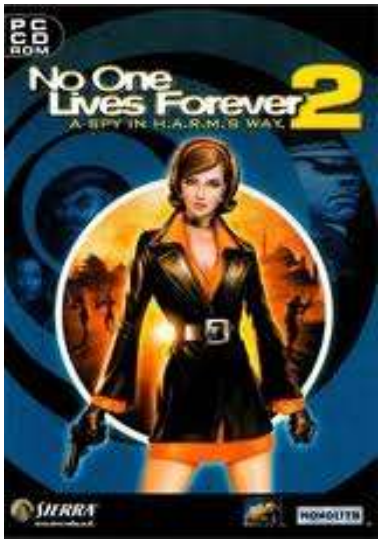
- Actions/behaviours associated with **states**
- States connected by **transitions** with associated **conditions**

Shooters: decision making example

- **Emergent behaviour** different in each play through in «*Sniper elite*» (Rebellion, 2005):
  - Range of finite state machines operating on waypoints of the level
  - Behaviours dependent on actions of other characters & tactical situation at nearby waypoints
  - A bit of randomness in the decision making process added «credibility» and an apparent cooperation (but no squad-based AI was there!!)

Shooters: decision making example

- Autonomous AI in «*No one lives forever 2*» (Monolith Production, 2002) was achieved by blending **FSMs** and **goal-oriented behaviours**:
  - Each character has a pre-determined set of **goals** which influence its behaviour
  - Periodically, the character evaluates goals to select the most relevant in that moment, which takes control of its behaviour
  - Inside each goal there is a FSM
  - Waypoints are used to check wether the character is in the correct position for a behaviour (firing, using a PC, etc.)
  - Waypoints nearby are used by the character to evaluate which actions are available



Shooters: perception

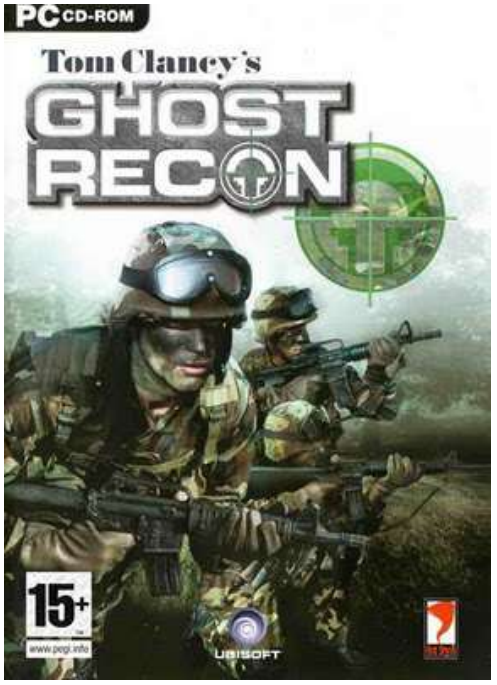
- May be faked by putting a **radius** around each enemy: it'll «come to life» whenever player enters the circle (e.g. Doom)

- More sophisticated aproaches may require a «real» **sense management system** …

Let's see several examples …

Shooters: perception example

- In «*Ghost recon*» (Red Storm, 2001 – NB: multiplayer!) the sense management system informs the AI characters about:
  - Amount of broken cover provided by bushes & c.
  - Amount of camouflge by testing character against its background
    (the line of sight goes straight to the character and then beyond it, till the next collisions: the id of the two materials are compared to determine the level of concealment)



Shooters: perception example

- «*Splinter cell*» (Ubisoft, 2002 - single player):
  - Each AI checks if the player is visible, taking into account dynamic mist, shadows, etc., but the background is **not** taken into account
  - When the concealment level drops under a certain treshold the player is spotted
  - AI also uses a cone-of-sight and a sound model to spot players

Shooters: pathfinding and tactical AI

- Developers use a wide range of representations for pathfinding, e.g.:
  - **(Waypoints)**
  - **Navigation meshes** (internal spaces), also known as «navmeshes»
  - **Hierarchical pathfinding**

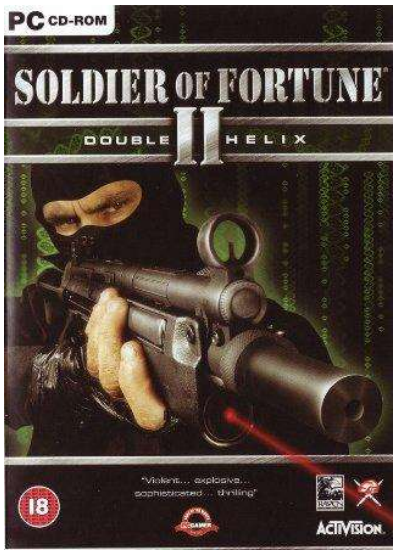
Let's see several examples …

Navigation mesh

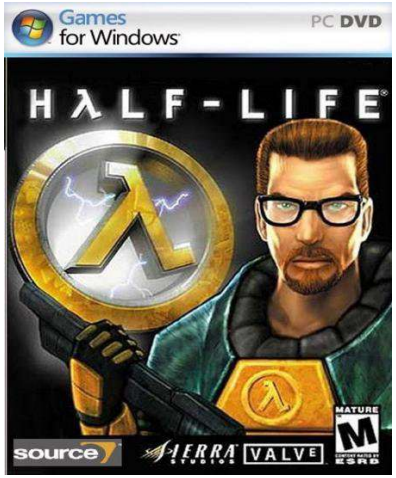
The graph is based on polygons: each polygon is a node, that may be interconnecetd to neighbouring ones.
Generally polygons are triangles/quadrilaterals => 3 o 4 interconnections at most

Shooters: pathfinding and tactical AI examples

- «**Embedded navigation**»: in *Soldier of Fortune* (Raven, 2002) links in the pathfinding graph are marked with the type of action needed to traverse them. Gives the illusion the agent reacts to the type of terrain

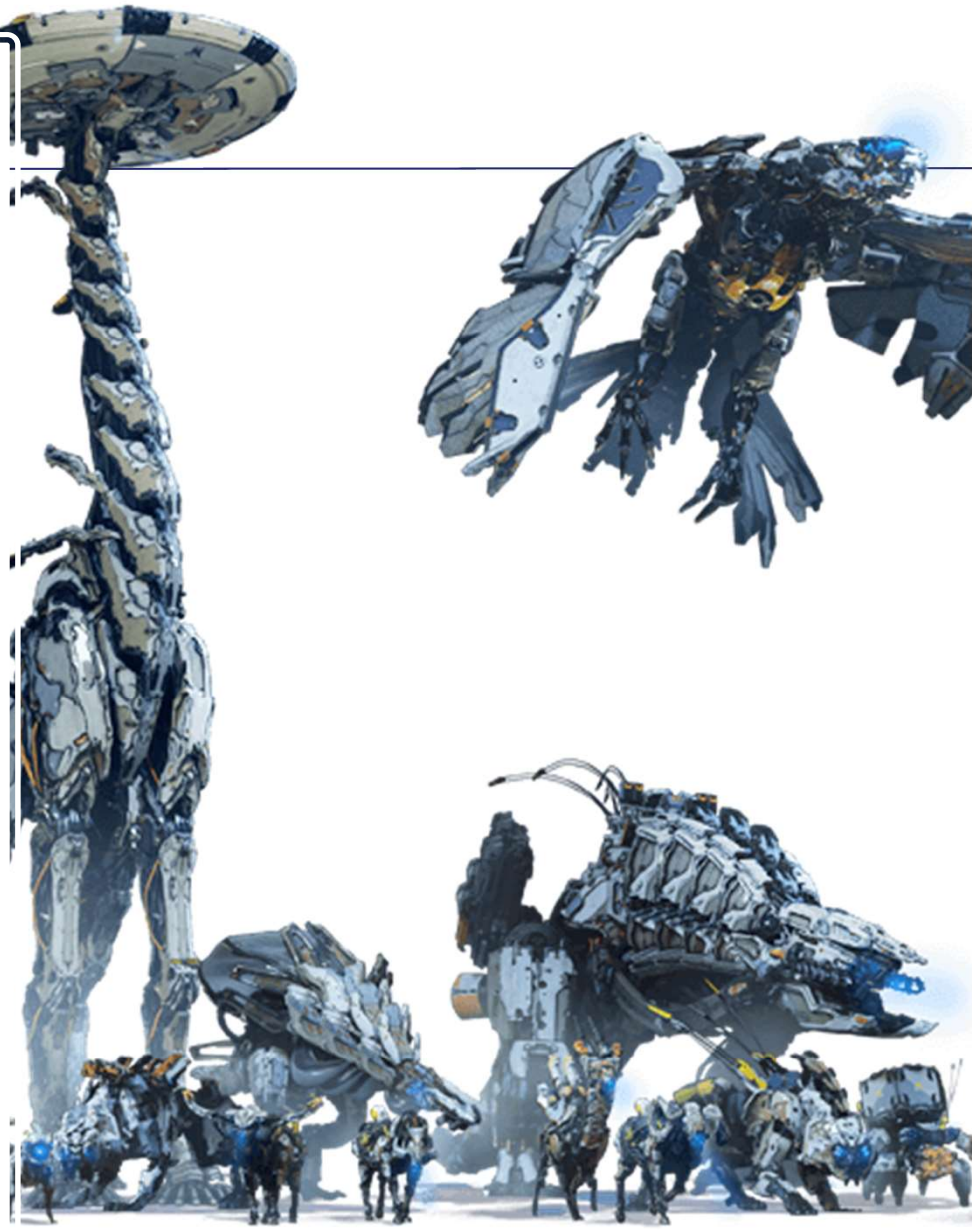


- **Waypoints**: in *Half-life* (Valve, 1998) the AI uses waypoints to figure out how to surround the player



Shooter-like games

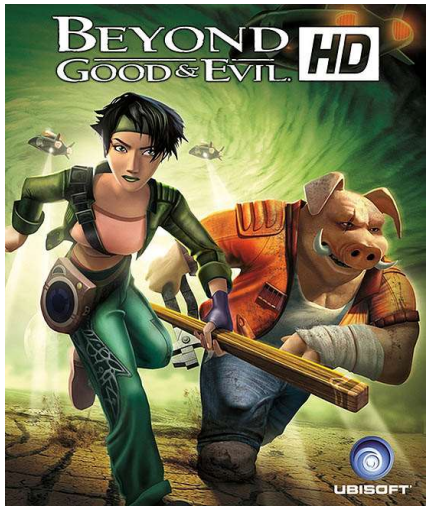
- A number of genres (not shooters) share similar characteristics with shooters:
  - Single player
  - First/third person view
  - Presence of enemies
- They use very similar AI tecniques



Shooter-like games: Platform and Adventure

- **Platform**:
  - (usually) younger audience
  - Make enemies interesting but fairly predictable

- **Adventure**:
  - Enemies = puzzle to solve (e.g. discover when an enemies lowers it shield, etc.)



Shooter-like games: Platform & Adventure

- Movement similar to FPS, but:
  - In platform also flying enemiers (2½D or 3D movement algorithms)
  - In adventure lots of animations to communicate character behaviour
- Pathfinding: generally not necessary
- Decision making:
  - *normal* (patrolling, random movements, set of animations) vs *spotted the player* behaviour (seek/pursue)

Shooter-like games: MMOGs/MOBAs

- Server is <u>always</u> running on a dedicated hw (it MUST be «protected» from the players …)
  - More resources …. GOOD !! …. ???

- Marginal need for AI (the challenge are other players), but in very large environment and for very large quantities of agents … BEWARE OF SCALABILITY ISSUES for **pathfinding** and **perception**
  - Pathfinding: pool planners, hierarchical pathfinding, instanced geometry (A* simply won't work with MANY mobs …)

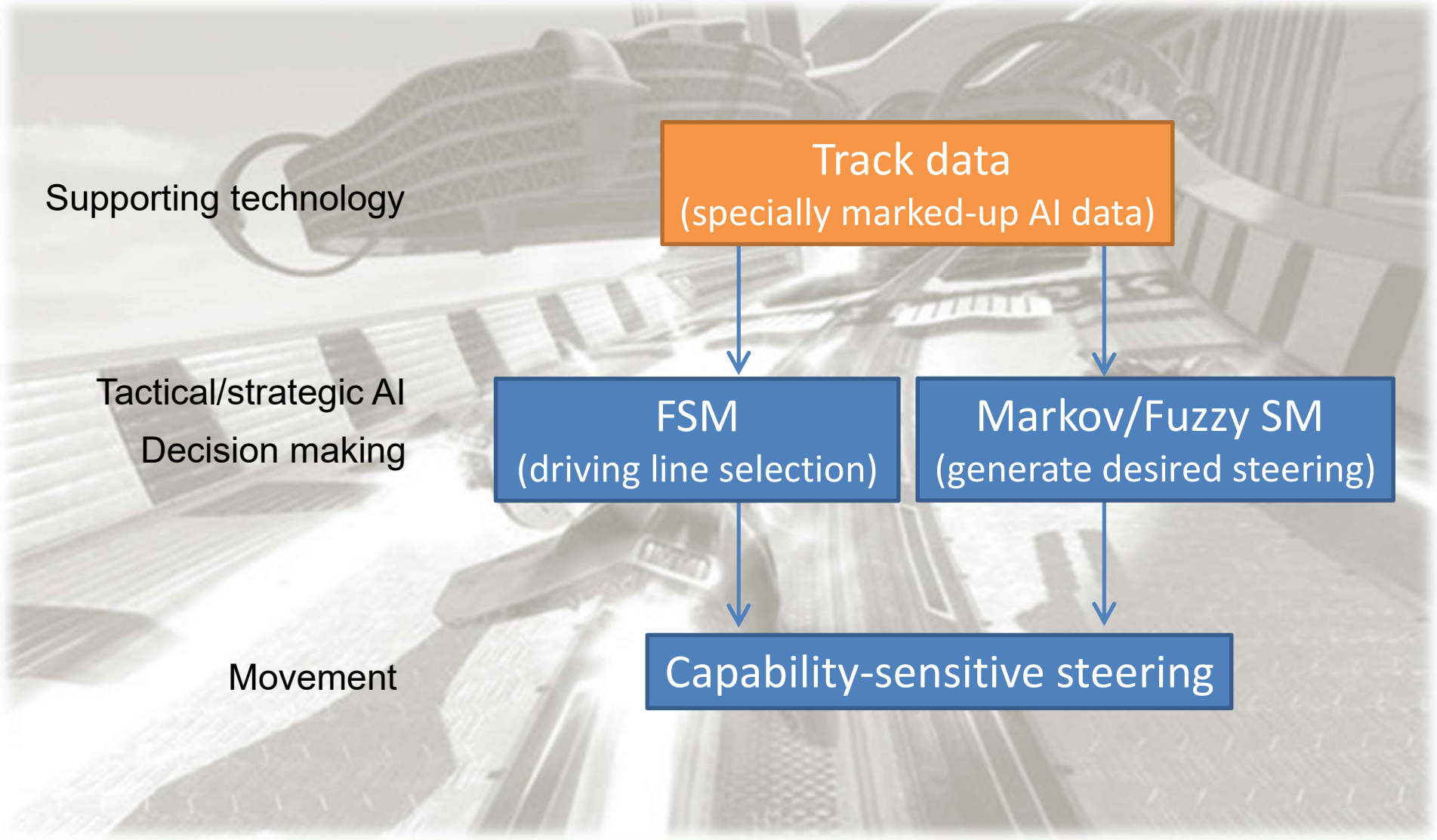
Designing games AI

AI and Driving games

Main AI needs for driving games

- One of the most specialized, genre-specific AI:
  - Crucial tasks focused on movement
- Player will judge the AI on the basis of how well it drives the car:
  - Generally realistic goal-seeking behaviour, clever tactical reasoning/route finding …. are NOT necessary
- Two main sub-generes:
  - Race driving
  - Urban driving

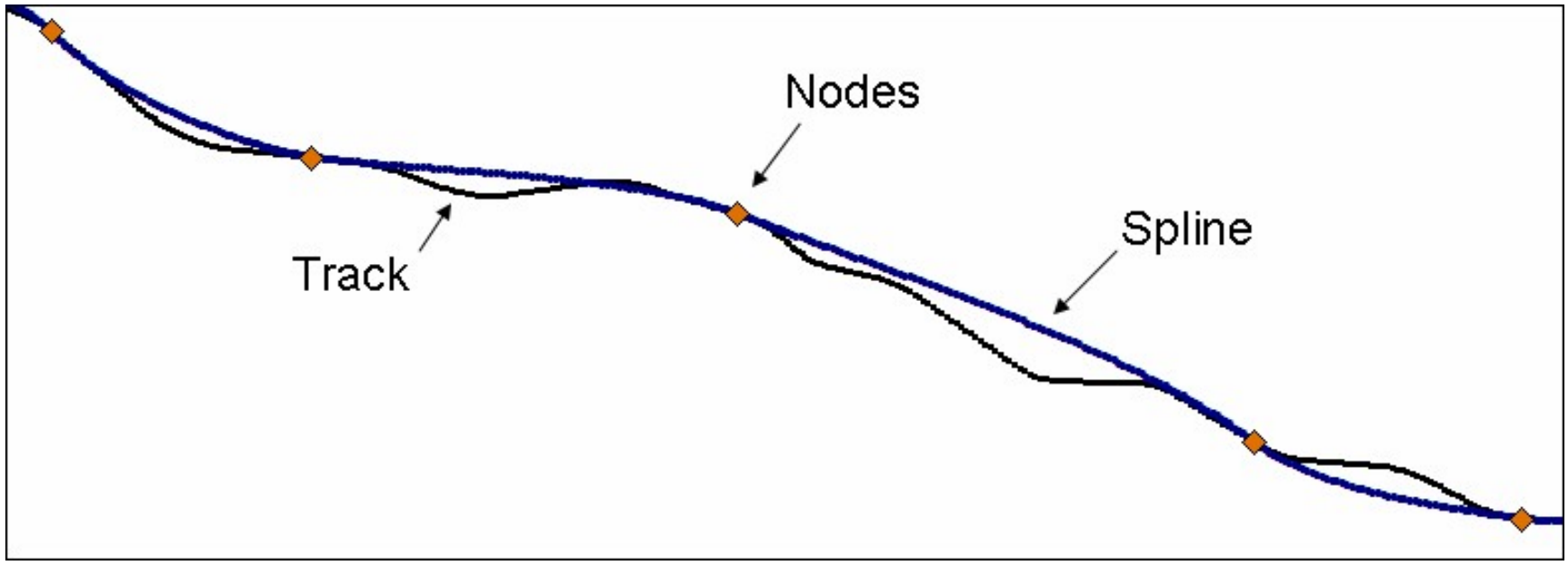
AI architecture for *race* driving games



Driving games: racing games movement

- Two main options (+ many variations …):
  - **Racing lines** (*splines* with data attached): no steering behaviour needed, level designer defines the spline and data (speed, steering to get back to the line if needed, etc.). Examples:
    - *Formula 1* (Bizarre 1997)
    - *Grand Theft Auto 3* (DMA 2001) for the cars in the background
  - **AI drives the car**: more recent approach, may be coupled with splines

Spline



In maths, a spline is a function, that is piecewise-defined by polynomial functions, and which possesses a high degree of smoothness at the places where the polynomial pieces connect (which are known as *knots*).

Driving games: racing games movement

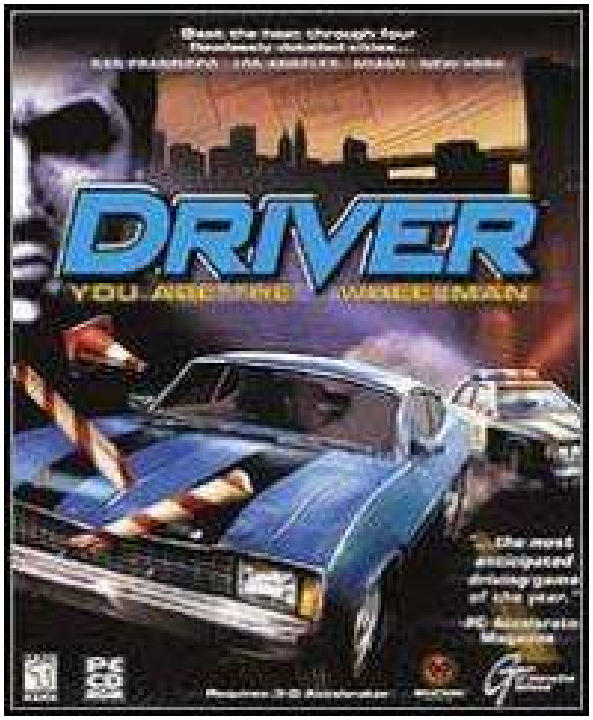
- **AI drives the car** - more recent approach:
  - Movement can be +/- realistic, according to the player expectations
  - Physics should be THE SAME for AI & player (critical issue!)
  - may be coupled with splines: AI tries to follow the line (always manually designed), but it is not «on a rail»
  - Special steering behaviour for overtaking: on a long straight (slower cars) vs on corners (F1)
  - Approach «chase the rabbit»

Driving games: special approaches to movement

- *Manic Karts* (Manic Media, 1995):
  - Fuzzy decision making instead of racing lines
- *Forza Motorsport* (Turn 10, 2005):
  - Neural networks to learn how to drive by observing humans
  - The shipped game was the results of hundreds of hours of training … !

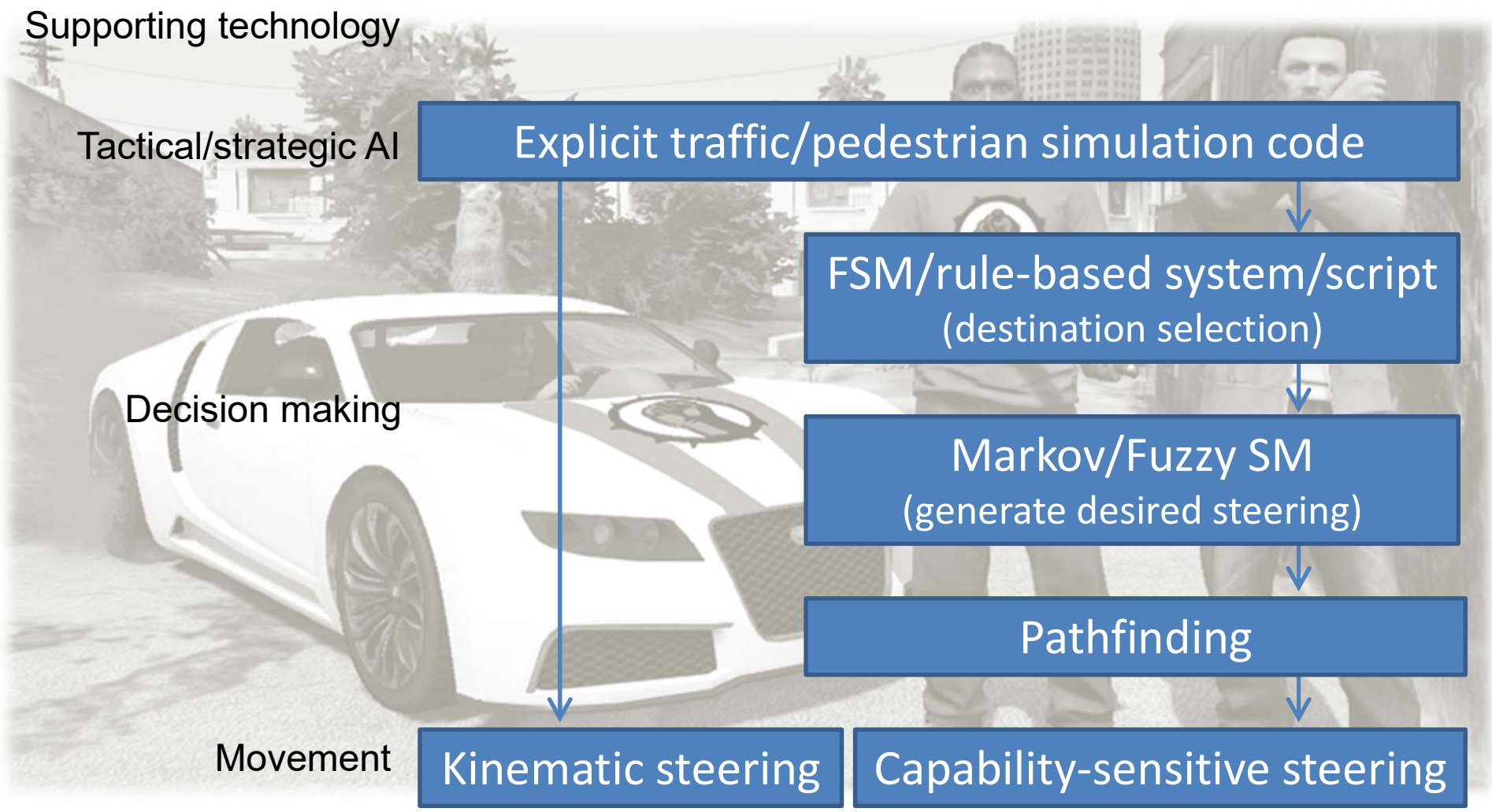
Driving games (urban): pathfinding and tactical AI

- *Driver* (Reflections Interactive, 1999) introduced a new genre:
  - Set in city, goal is to avoid/catch other cars
  - NO predefined fixed racetracks!
  - Enemies catching the player



- solutions:
  - AI follows path for escaping / performs homing-in algorithm for catching player
  - Cars created only in the surrounding block (more added when needed)
  - Pathfinding for catching the player
  - Tactical analysis for escaping routes or to surround the player

AI architecture for *urban* driving games



Driving-like games

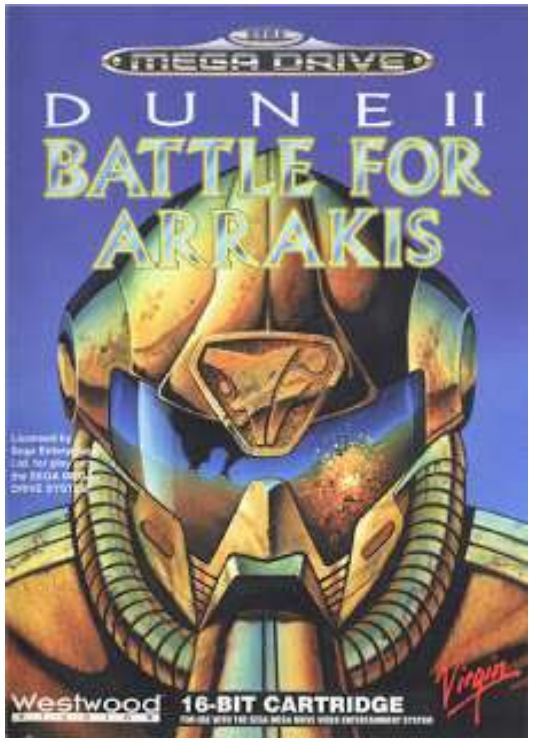
- The same approaches used for driving games can apply to:
  - **Extreme sports games** (*SSX*, EA 2000; *Downhill Domination*, Incog 2003):
    - Racing line + tricks sub-game (e.g. a combo to be performed during jumps)
  - **Futuristic racers** (*Wipeout*, Psygnosis 1995)
    - Racing + weapons
    - Aiming and firing system
    - Decision system (slow down to be overtaken and target the player, etc.)

Designing games AI

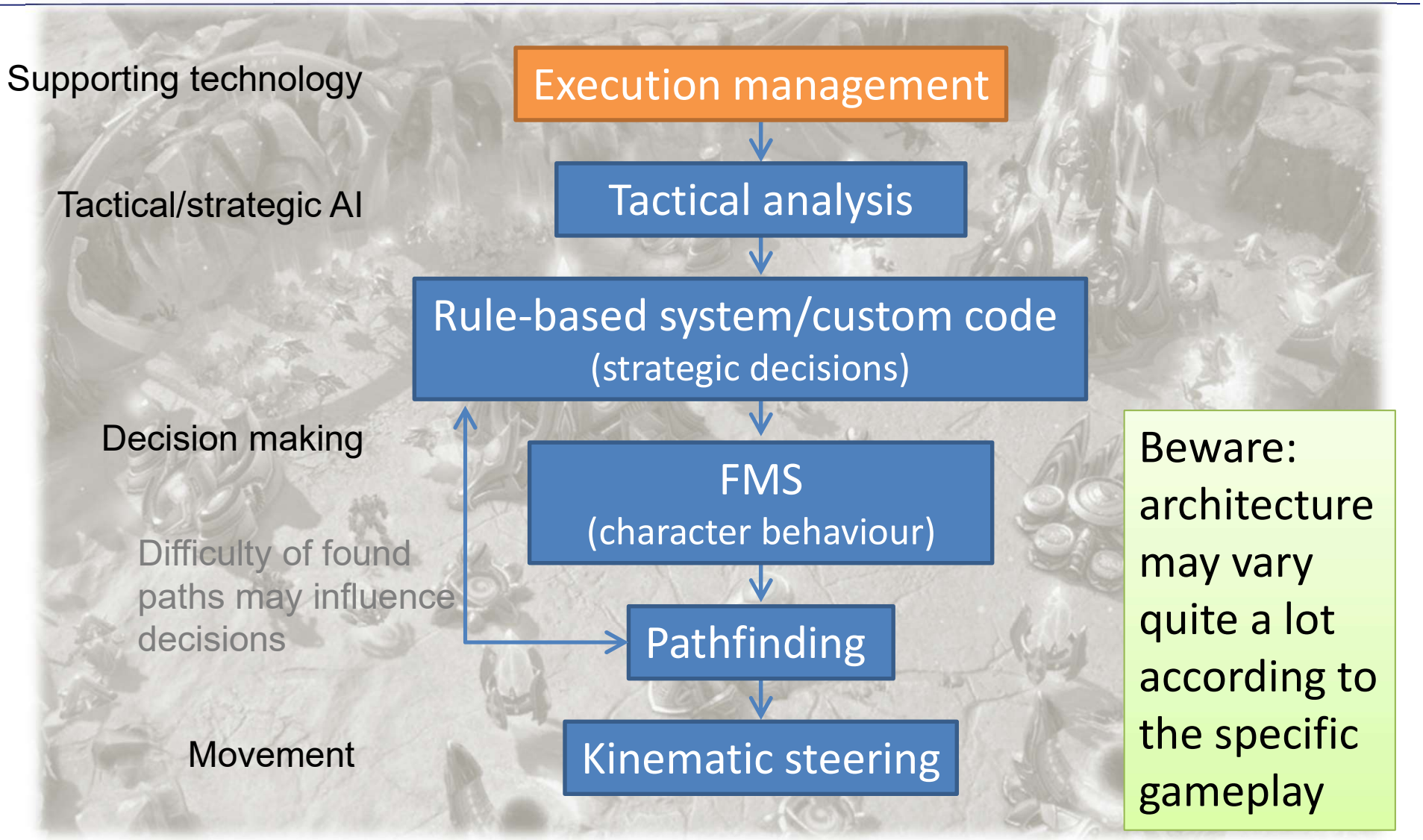
AI and Real Time Strategy games

Main AI needs for RTS games

- *Dune II* (Westwood, 1992): not the 1° RTS, but created a new genre (more diffused on PCs), whose key needs for AI are:
  - Pathfinding
  - Group movement
  - Tactical and strategic AI
  - Decision making

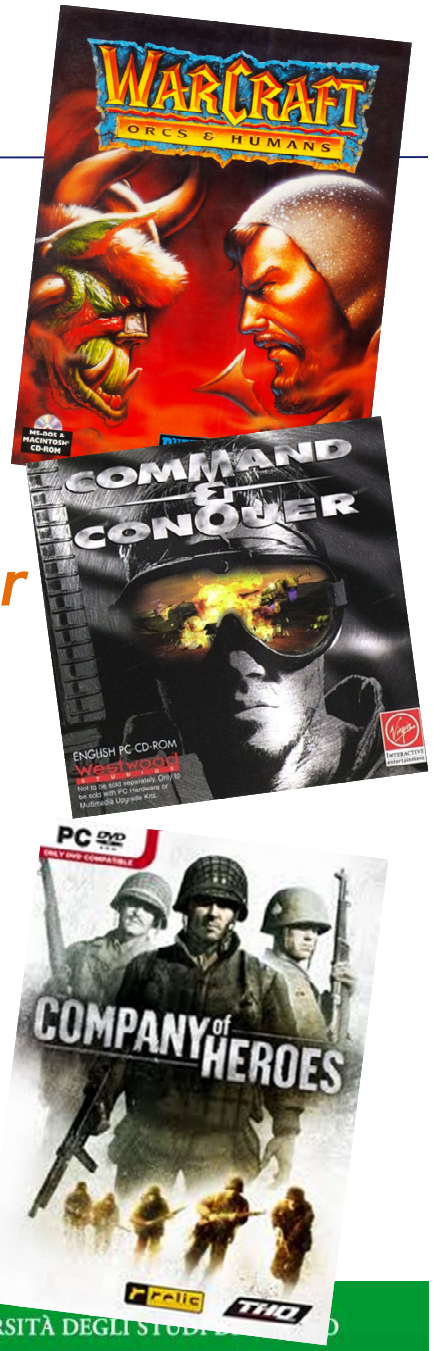


AI architecture for *RTS* games (to be personalized)

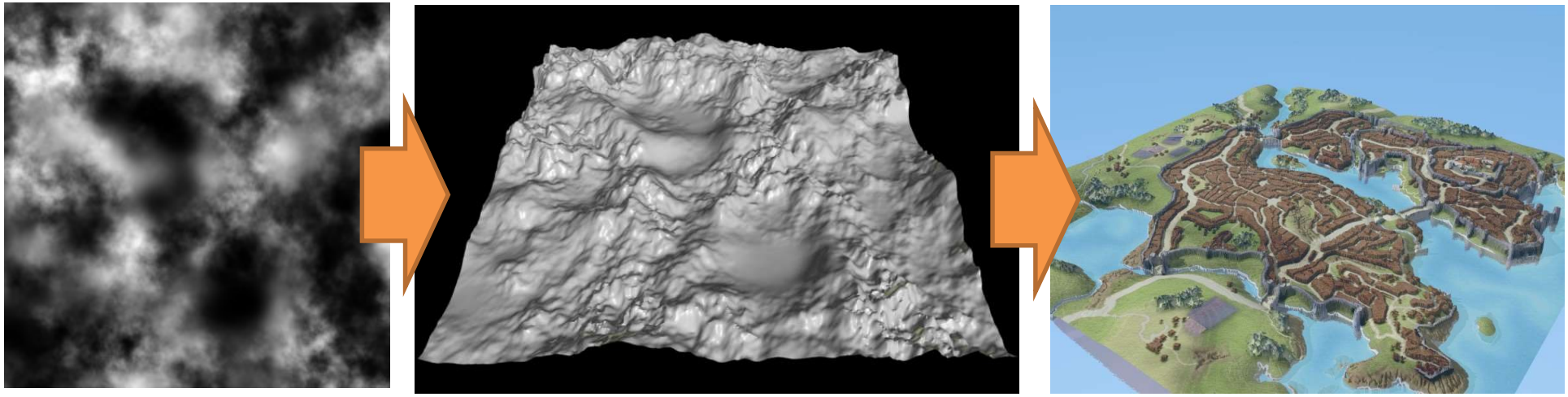


RTS games: pathfinding

- Early RTS (*Warcraft: Orcs and humans*, Blizzard 1994; *Command and Conquer*, Westwood, 1995) were based on huge tiled grids:
  - Focus on **efficient and quick pathfinding**
- Recent RTS use arrays of heights (***heightfield or heightmaps***) to render the landscape, that are used also for pathfinding (in some case pre-processed)
- Several RTS use also ***deformable terrain*** (*Company of Heroes*, Relic, 2006 – bombs may destroy buildings, create craters, etc.), increasing pathfinding difficulties (no pre-computation possible)



Heightfield (or Heightmaps)



- Heightfield are raster images used to store values:
  - surface elevation data, for display in 3D
  - bump mapping to calculate where 3D data would create shadows
  - displacement mapping the heightmap is converted into a 3D mesh
- widely used in terrain rendering sw and video games
- ideal way to store digital terrain elevations: require substantially less memory than a regular polygonal mesh

RTS games: group movement

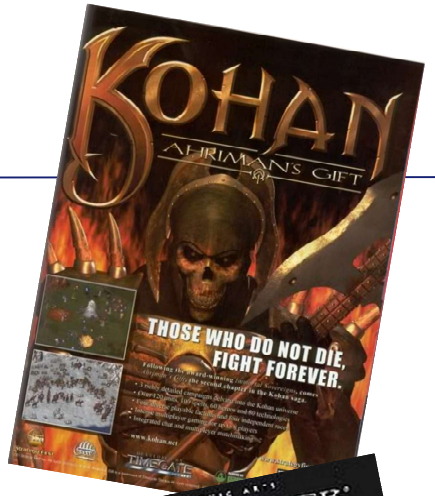
- Main issue: **group movement**

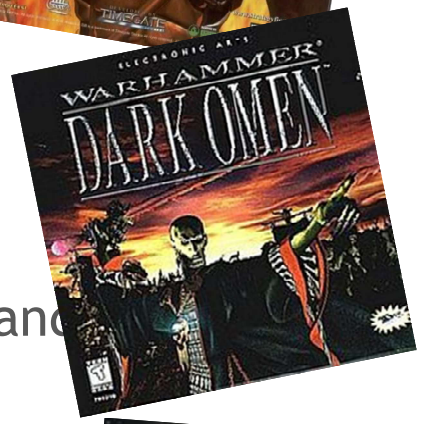
- Older RTS:
  - *Kohan: Ahriman's gift* (TimeGate 2001) and *Warhammer: dark homen* (Mindscape 1998):

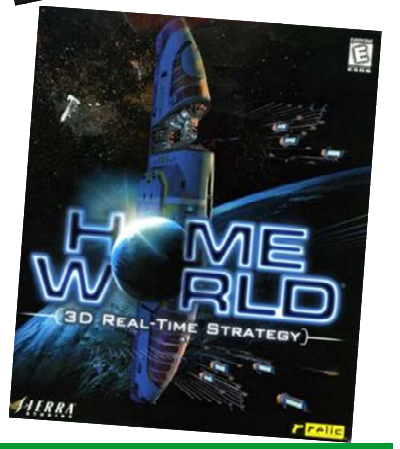
    units are collected into formations (with size limit) and moved has a whole. Motion-system has pre-defined patterns

  - *Homeworld* (Relic 1999):

    formation extended in 3D, no size limit (need for scalability) => scalable formation with different slots according to group numerosity







RTS games: group movement

- Main issue: **group movement** management

- More recent RTS:
  - *Full Spectrum warrior* (Pandemic 2004):

    formation depends on the features of the level (single line near a wall, double behind a cover, etc.).
    Player has only indirect control on the group: formation pattern decided by the AI Units move independently and cover each other if needed

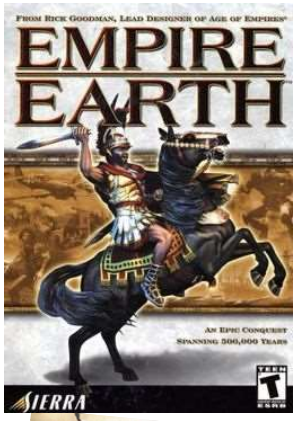
RTS games: tactical and strategic AI

- RTSs introduced **influence mapping**

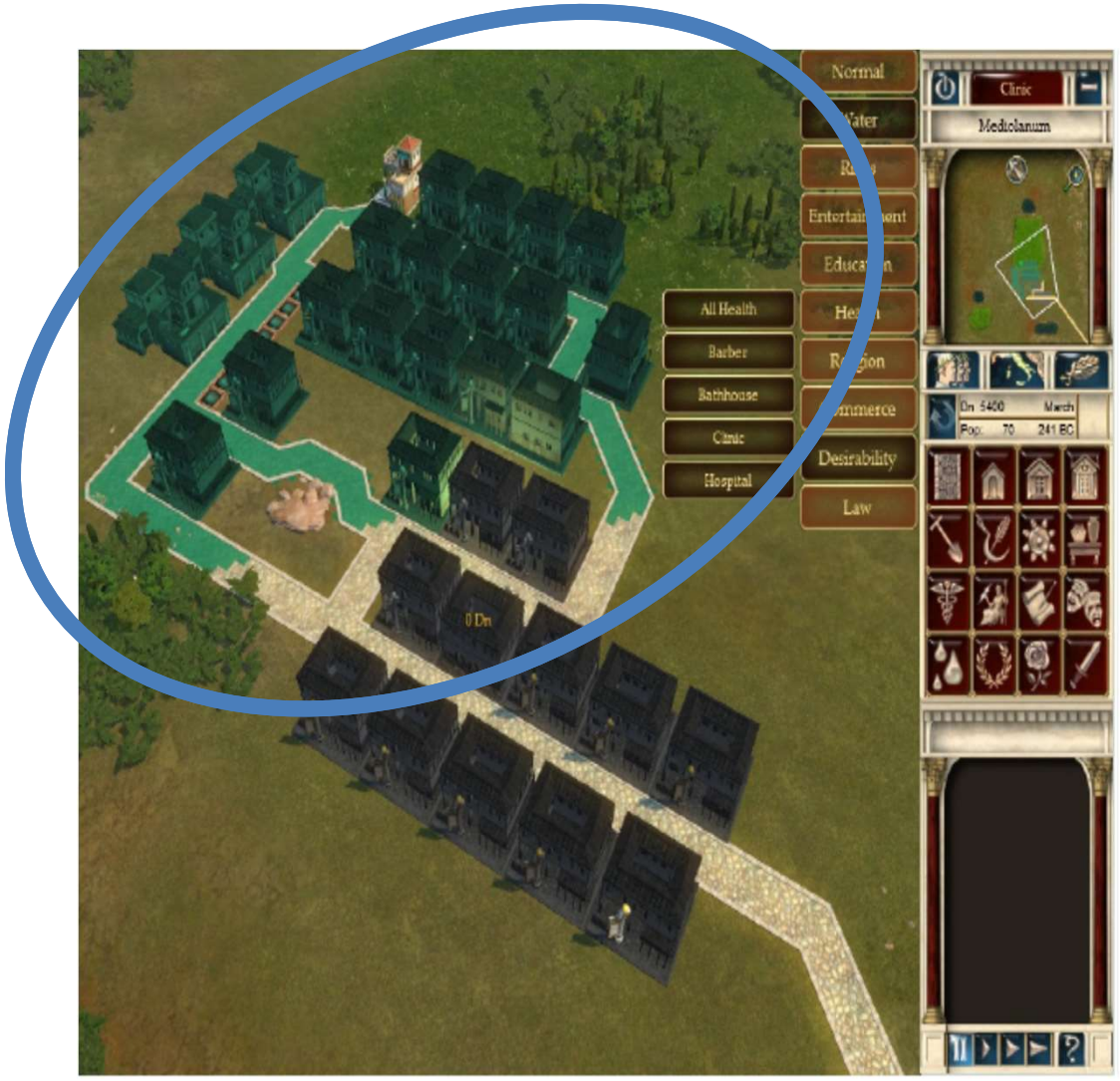
WARCRAFT

STARCRAFT

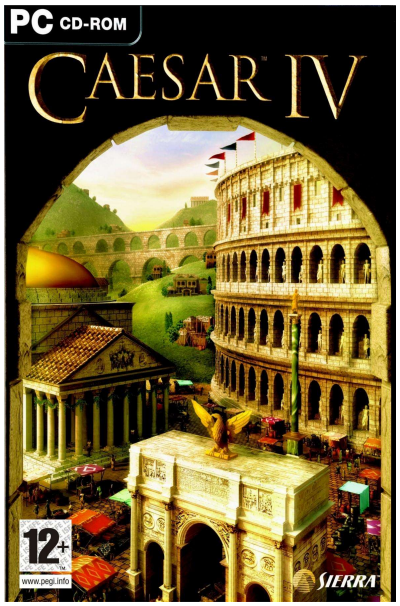
- Output of tactic/strategic AI used for:
  - Guiding pathfinding: units in *Total Annihilation* (Cavedog 1997) took - for the 1° time - into consideration terrain (e.g. rocky formation) when moving
  - Selection of location for construction, but city walls are an open issue (often pre-designed by a level designer): in *Empire Earth* (Stainless Steel, 2001) AI used influence mapping + spatial reasoning to put walls between valuable areas and enemies
  - Large scale troop maneuvers: in *Empire: total war* (The Creative Assembly, 2009) AI directs units towards the enemy and tries to avoid cannons and attacks



Influence map: an example



- *Caesar IV*: clinic coverage for a neighbourhood



RTS games: decision making

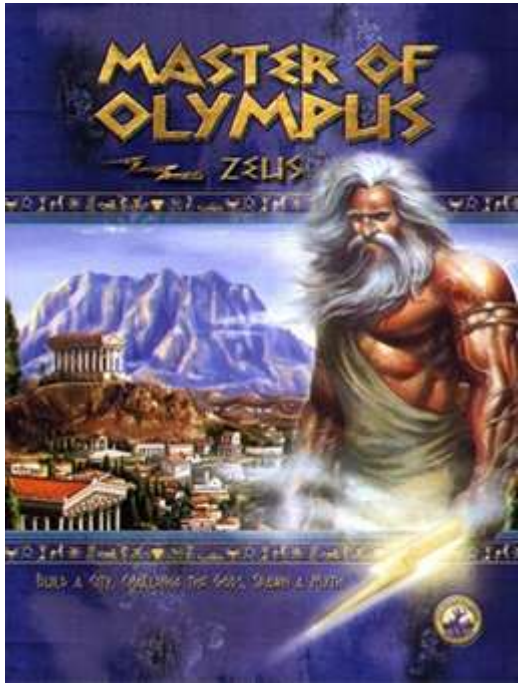
- Multiple AI levels for decision making:
  1. individual unit
  2. group
  3. whole side

For each level an AI component is created. Complexity varies from game to game

- AI modules can be or not independent (e.g. in *Warcraft 3: Reign of Chaos* the central AI may decide to play in an offensive style)
- Decision making technology: **FSMs, decision trees, Markov & probabilistic methods, set of rules**

Example: Zeus master of Olympus (Sierra 2000)

Multiple AI levels for decision making:



1. individual unit:
   - Heroes, gods, monsters
2. Groups:
   - groups of citizens, merchants
3. whole side:
   - Enemy's armies

Designing games AI

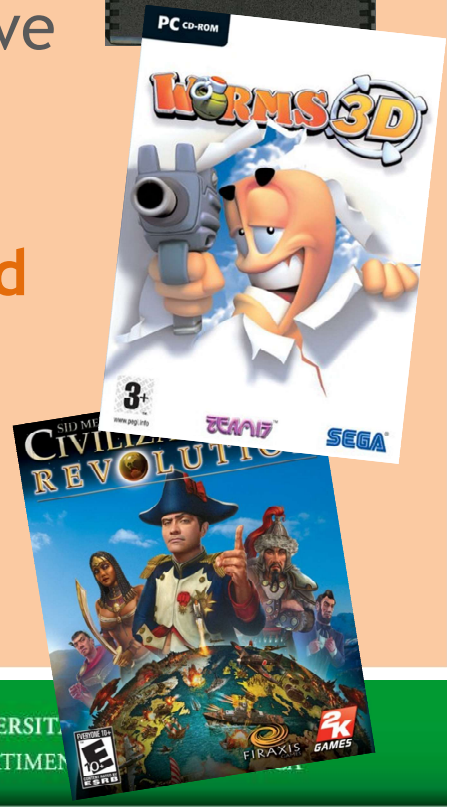
AI and Turn-based Strategy games

Main AI needs for turn-based strategy games

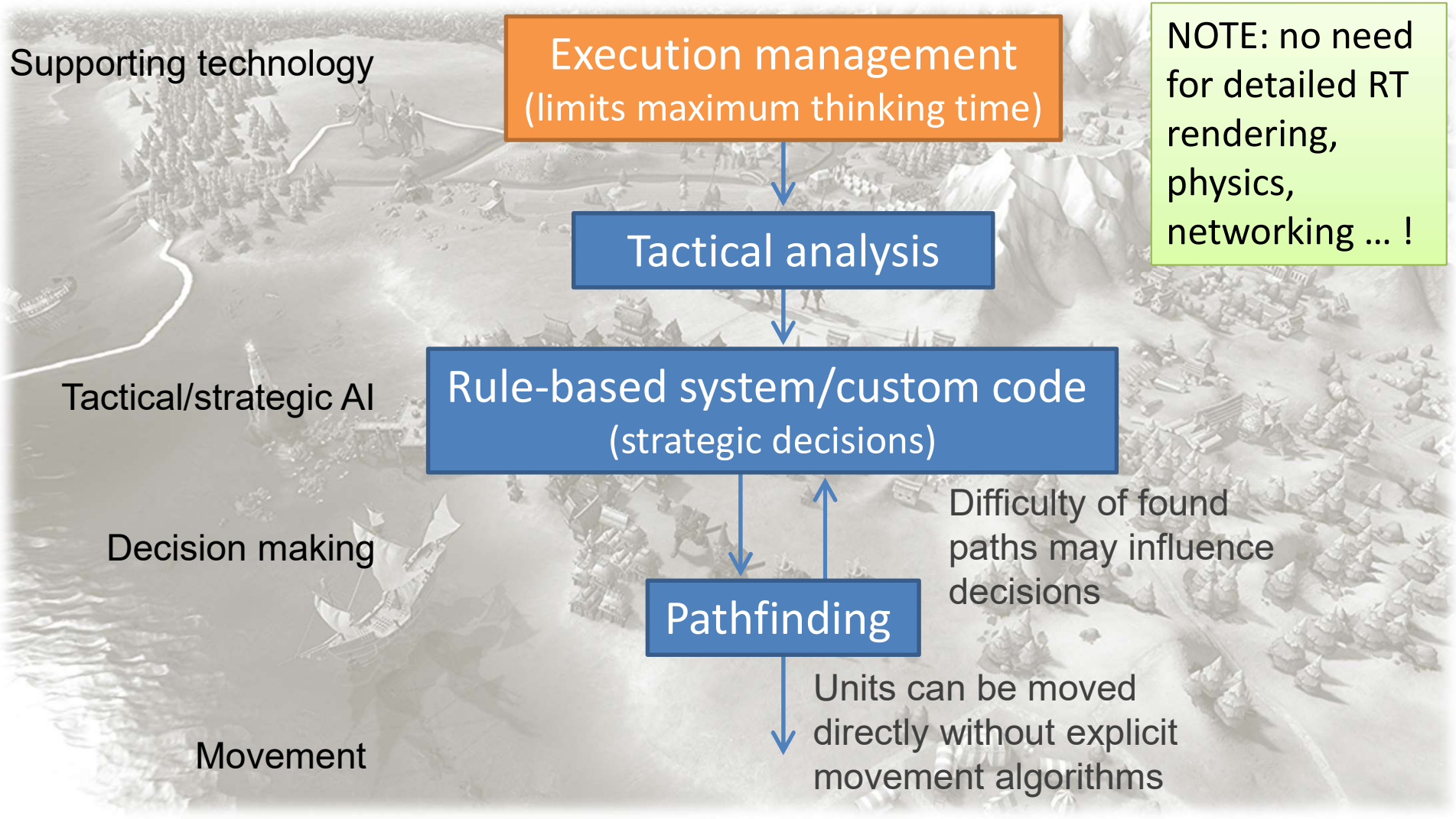
- Older turn-based strategy games:
  - Variant of existing board games (*3D Tic-Tac-Toe* - Atari 1980, *Computer Bismark* – Strategic Simulations 1980)
  - Minimax techniques were often enough

- More recent, highly complex turn-based strategy games (*Cid Meier's Civilization*, *3D Worms*) have almost unlimited number of possible moves => needs similar to RTS:
  - ***Pathfinding, decision making, tactical*** and ***strategic*** AI,

    but
  - Quite simple movement algorithms (kinematic, position update, etc.)



AI architecture for *turn-based strategy* games



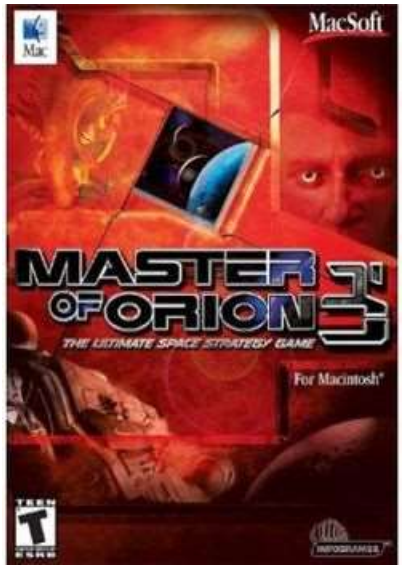
Turn-based strategy games: timing

- Both player and computer have longer time to think than in RTS, but
  AI has to compete with a higher level of human thinking

  - Game/level designers must try to simplify as much as possible the AI job, by, e.g. creating levels easy to tactically analyze, research trees easily searchable, etc.
  - Beware of the full range of decisions to take: economic system, research tree, movement, resource managment, etc. etc.

Turn-based strategy games: helping the player

- TBSGs should also help the player to **automate boring tasks**:
  - Players of *Master of Orion 3* (Quicksilver, 2003) can assign a number of decisions to the AI

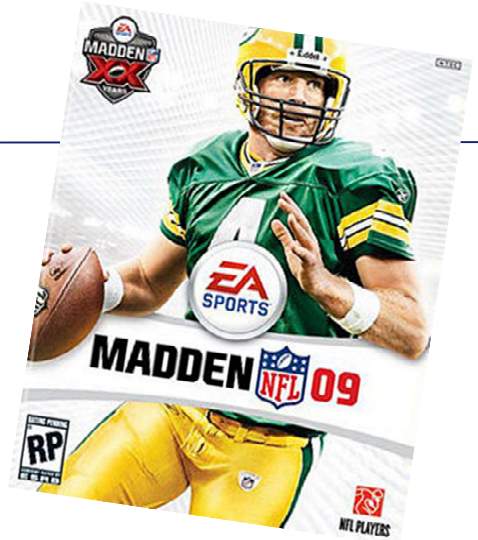


- An **assistive AI** implies:
  - having decision tools which have no (or little) strategic input from higher levels
  - Understanding the player strategy … !!!

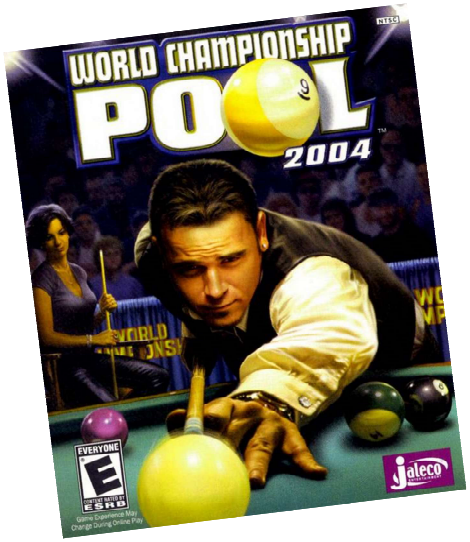
Designing games AI

AI and Sport games

Main AI needs for sport games

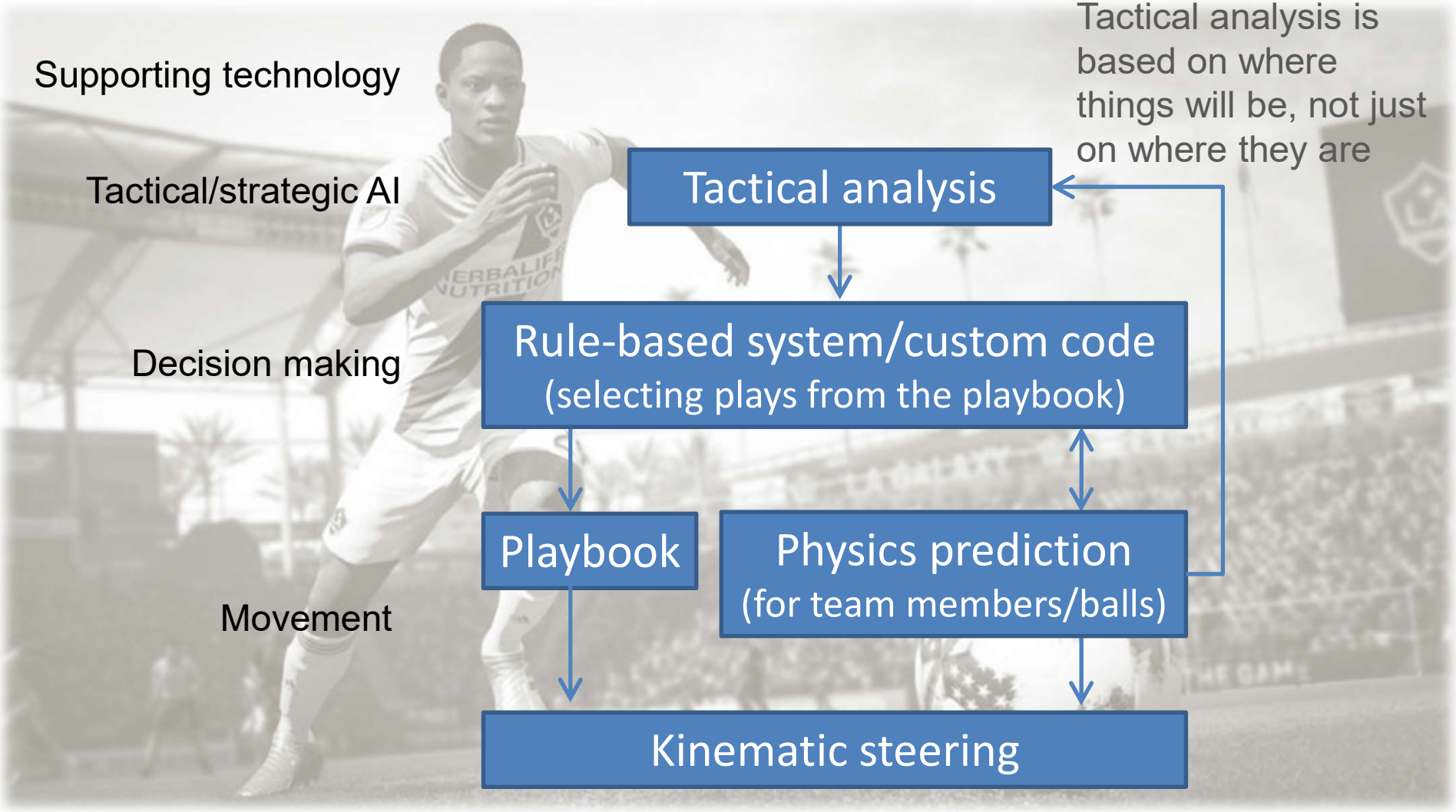


- Huge range of different games, e.g.:
  - *Madden NFL 2009*, EA 2008
  - *World Campionship Pool 2004*, Blade 2004
  - *FIFA* series



- Huge body of knowledge ready to use for generating good strategies, but:
  - Not always easy to encode
  - Players expect human-like compentencies
  - Team sports require appropriate team coordination
- Multi-level AI:
  - Higher level: strategic decisions
  - Middle level: coordinated motion system (play patterns)
  - Lowest level: single unit

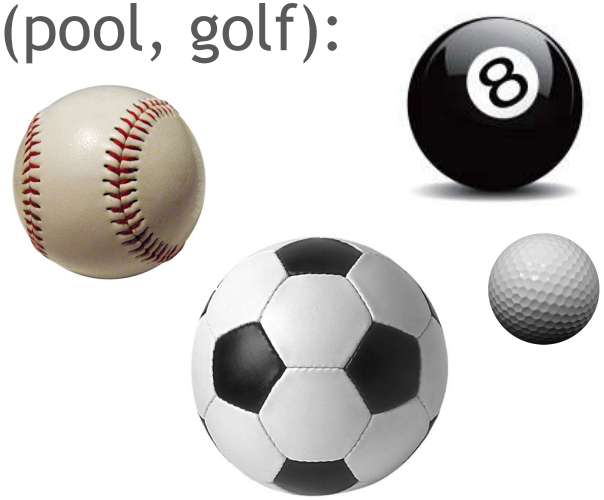
AI architecture for *sports* games



Sport games: physic prediction

- **Physic prediction**: balls movement must be predicted in order to allow AI to make decisions (e.g. intercepting the ball)

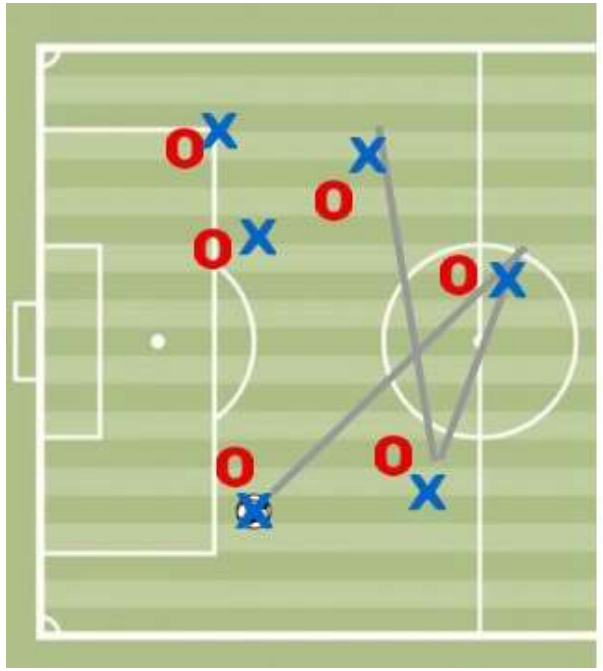
- Complex dynamics integral to the game (pool, golf):
  - Physics should predict the result
- Simpler dynamics (soccer, baseball):
  - Predicting the trajectory could be enough



Same solutions used for firing weapons can be used to predict ball movement

Sport games: playbooks & content creation

- Playbooks: set of movement patterns used by teams in specific circumstance. May refer to:
  - The whole team
  - Smaller groups of players



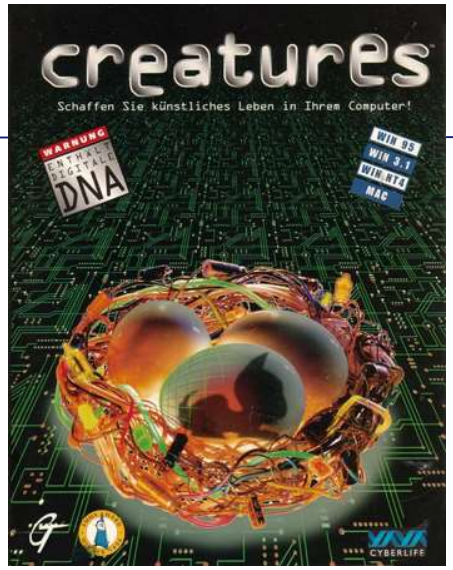
- Algorithms for making sure that characters move at the correct time
- Formation motion system

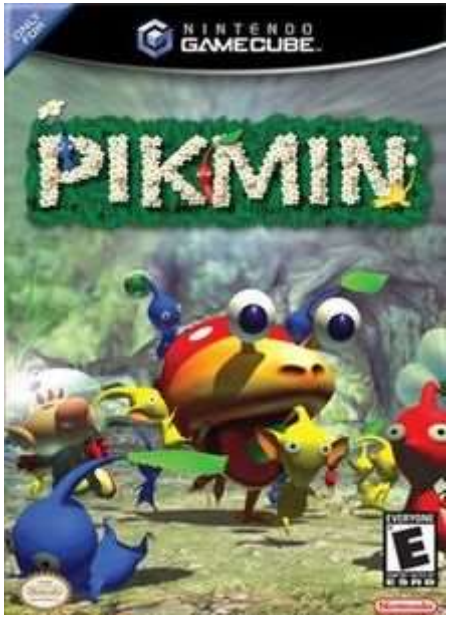
Designing games AI

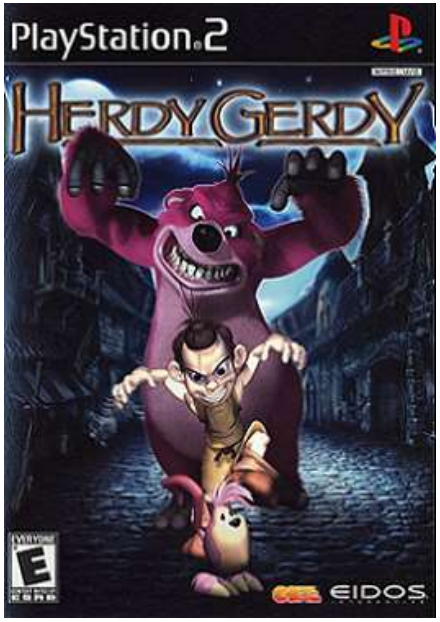
AI-based games

AI- based game genres

- Teaching characters
- Flocking and herding games







AI supporting game design

- We will see several examples during the course about the use of:
  - Genetic algorithms
  - Procedural content generation
  - Neural networks & machine learning
  - Etc.


TECHZAMAZING
ROBOT UNICORN ATTACK 3
FOREVER


GALACTIC
ARMS RACE


CLOUDBERRY
KINGDOM


SPORE
WIN
MAC
EA