An Action-Driven AI Architecture for Non-Player Characters in Video-Games

Davide Aversa

SAPIENZA
UNIVERSITÀ DI ROMA

Introduction

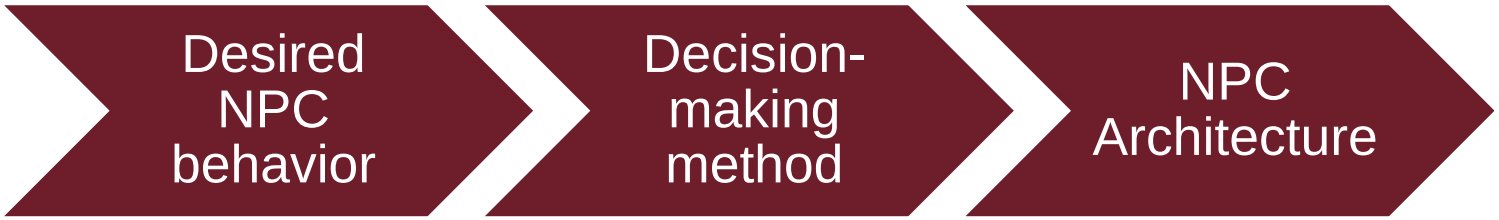
- In this thesis we focus on:
  - Artificial Intelligence in video games
  - Video games with non-player characters (NPCs)
  - How do NPCs decide about what action to do next (Action-based decision making)
  - How do NPCs sense, keep track of information, execute actions, etc (AI Architecture)

AI action-driven decision making for NPCs

- In the last years many decision making techniques has been developed in the video game industry (mostly inspired by AI research):
  - **FSMs** and **Hierarchical FSMs** (HFSM)
  - **Behavior Trees** (Halo 2, 2004)
  - **Goal-Oriented Action Planning** (F.E.A.R., 2005)
  - **Hierarchical Task Networks** (Killzone 2, 2009)
  - **Utility Based System** (The Sims 3, 2009)
- But what about the NPC architecture?

AI action-driven architectures for NPCs

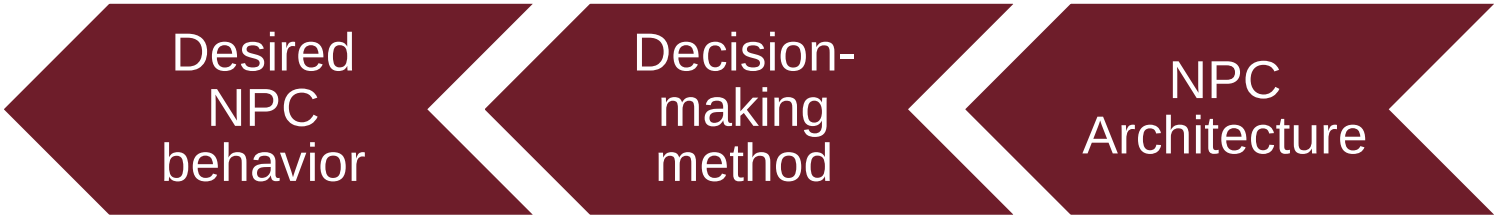
- Typically the NPC architecture is “built around” the decision making component that is used to model the behaviour of the NPC



- As a consequence, NPC architectures in games are more “game-specific” and less “general-purpose”.

Thesis objectives

- **Design** a general-purpose action-based AI architecture



- **Develop** the proposed architecture as a software package for a popular game engine
- **Validate** the approach by implementing NPC scenarios

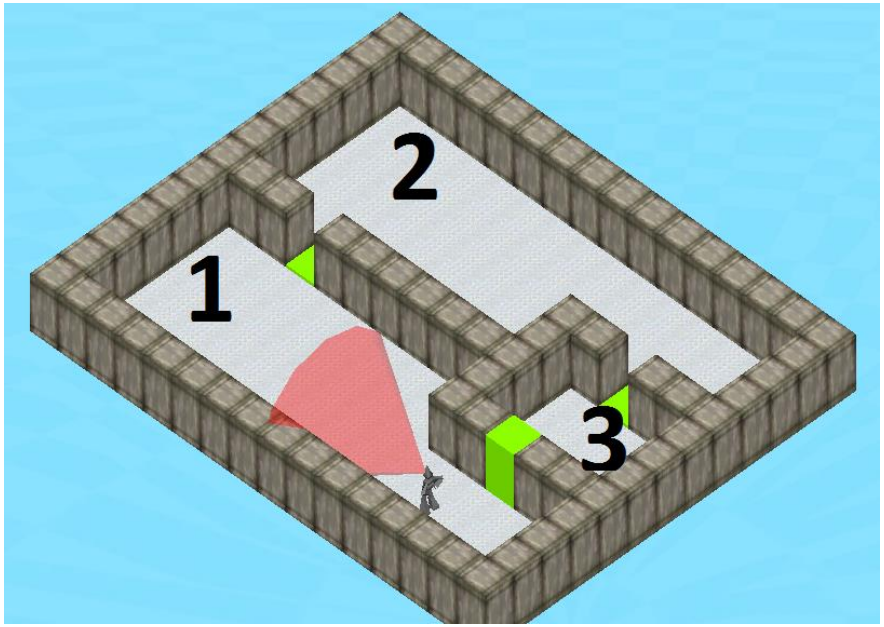
Thesis objectives

- **Properties** of the proposed architecture:
  - **Modularity:** the architecture is based on a minimal set of components which can be composed in order to develop complex practical configurations.
  - **Reusability:** the proposed components are decoupled from each other so that the same code can be used with multiple instances of a component.
  - **Context abstraction:** the architecture is independent from the AI technique used for decision making and the particular game setting.

Motivating example: Personalized pathfinding

- Typically in video games, pathfinding (compute a path to go from A to B) is handled by a single component.

- Every NPC asks for a path to the same pathfinding component.
- What if we want an NPC to navigate according to its knowledge of the map connectivity?



Motivating example: Personalized pathfinding

- Solution 1: Multiple instances of the pathfinding component.
  - Needs too many resources, also introduces much redundant information and computation!
- Solution 2: Separate high-level (personalized) from low-level (common) topology:
  - Each NPC knows the topology of the map
  - Each NPC has internal knowledge of the connectivity between areas

Motivating example: Personalized pathfinding

- We want to facilitate solutions like the 2nd alternative (and many more!) using a **general-purpose** architecture for NPCs that takes into account **sensing**, **thinking**, and **acting** in a **principled way**.

- The proposed CogBot architecture
  - Four **basic primitive components** with minimal interface
  - Many architecture instances as **complex configurations** of these components

CogBot architecture components

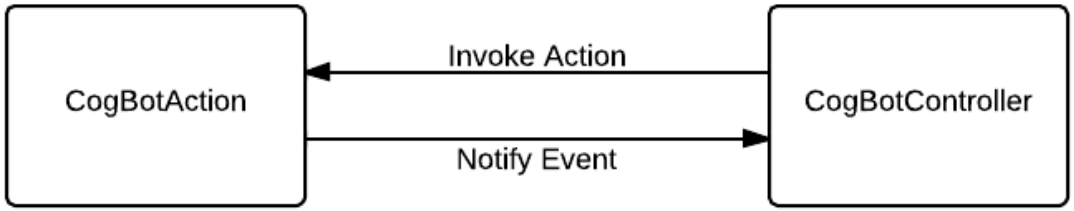
CogBotAction

CogBotPerception

CogBotControl

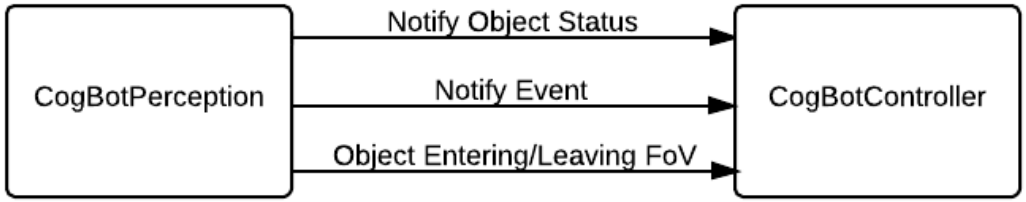
ICogBotDeliberation

CogBot architecture components: CogBotAction



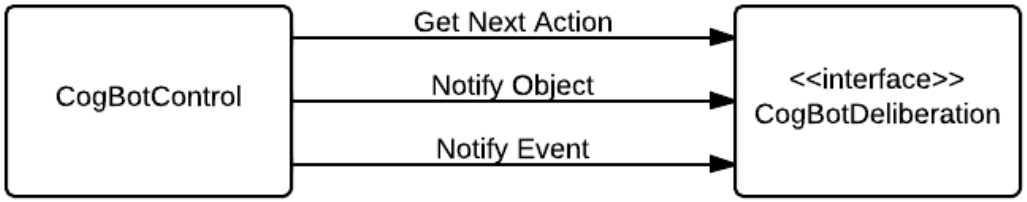
- *BotAction* component is used to decouple actions from the other components of the architecture.
- An action is invoked using the DoAction method contained in the component.
- An action is specified as a string containing the action name and a list of parameters.

CogBot architecture components: CogBotPerception



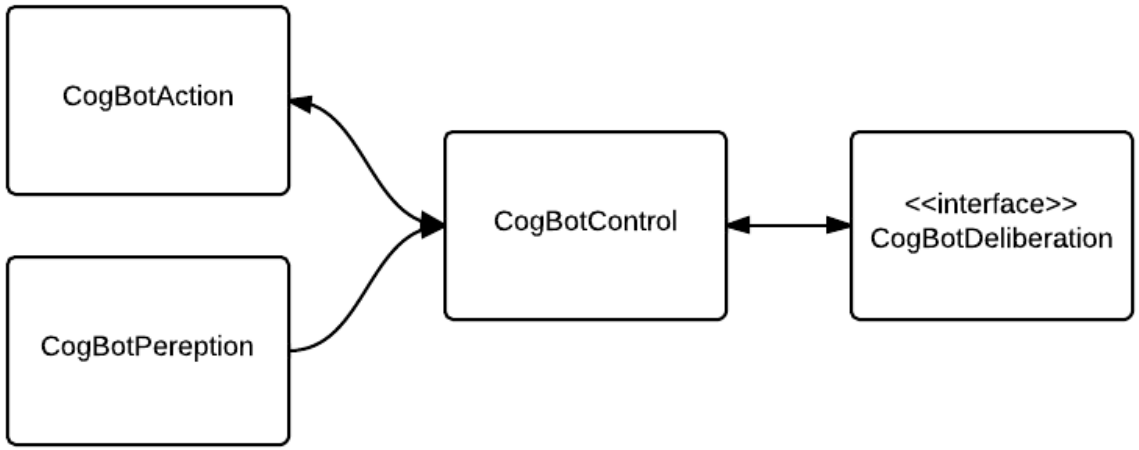
- *BotPerception* component adds the perception capability to any closed collider mesh.
- *BotPerception* is used to receive notifications about objects inside the perception mesh.
- It also automatically performs a visibility check for the objects in order to avoid to perceive objects hidden by other entities.

CogBot architecture components: ICogBotDeliberation



- *IBotDeliberation* is an interface for deliberator components.
- Every implementation of this interface is responsible for deciding the immediate next action that should be performed by the NPC.
- Can be used to abstract a number of decision making methods such as Behavior Trees, Finite State Machines, Goal-Oriented Action Planning, etc.

CogBot architecture components : CogBotControl

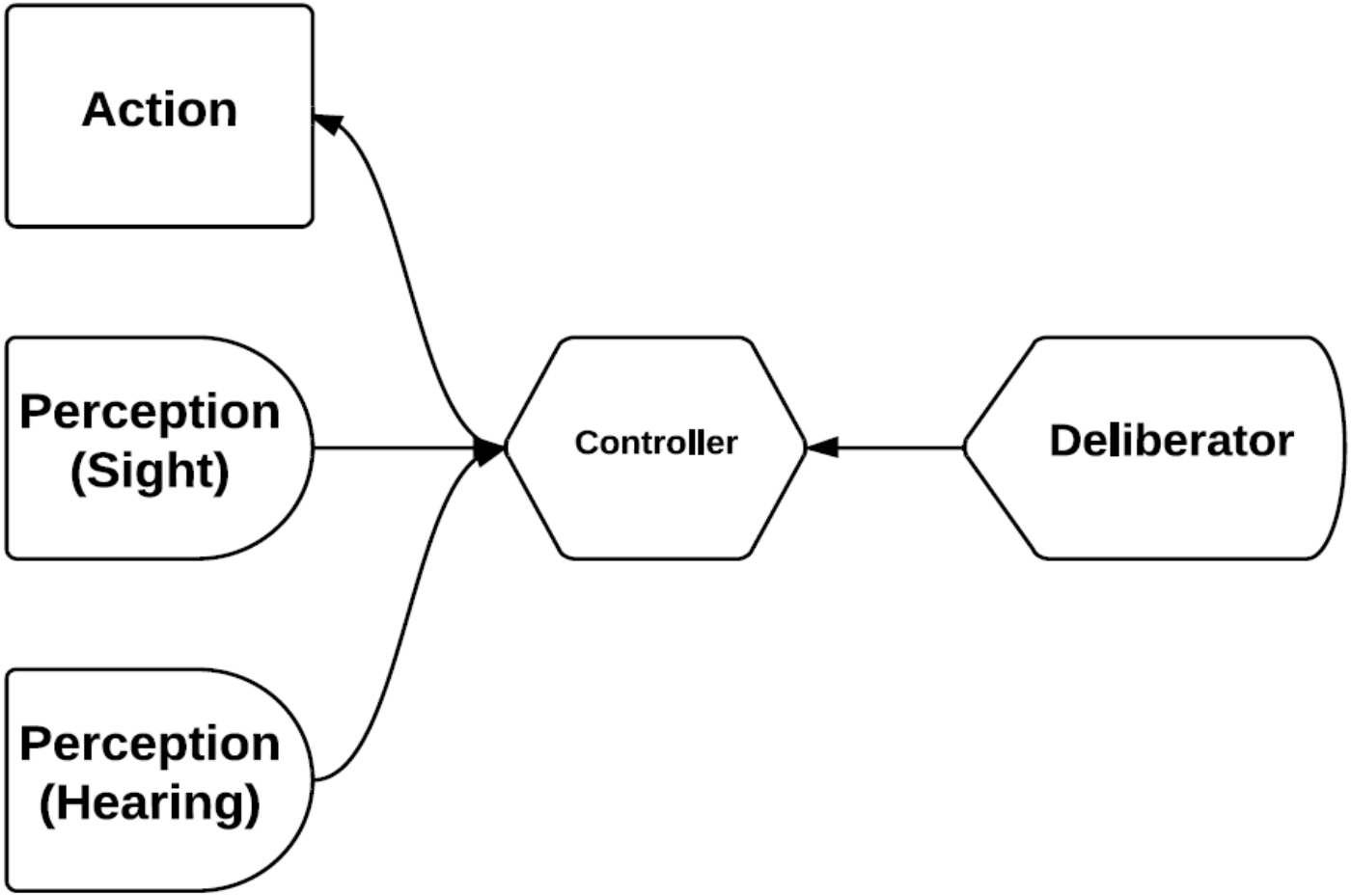


- *BotControl* is responsible for going over a loop that passes information between the perception and deliberation components, and handling the execution of actions as they are decided.
- It is agnostic of the way that perception, deliberation and action is implemented.

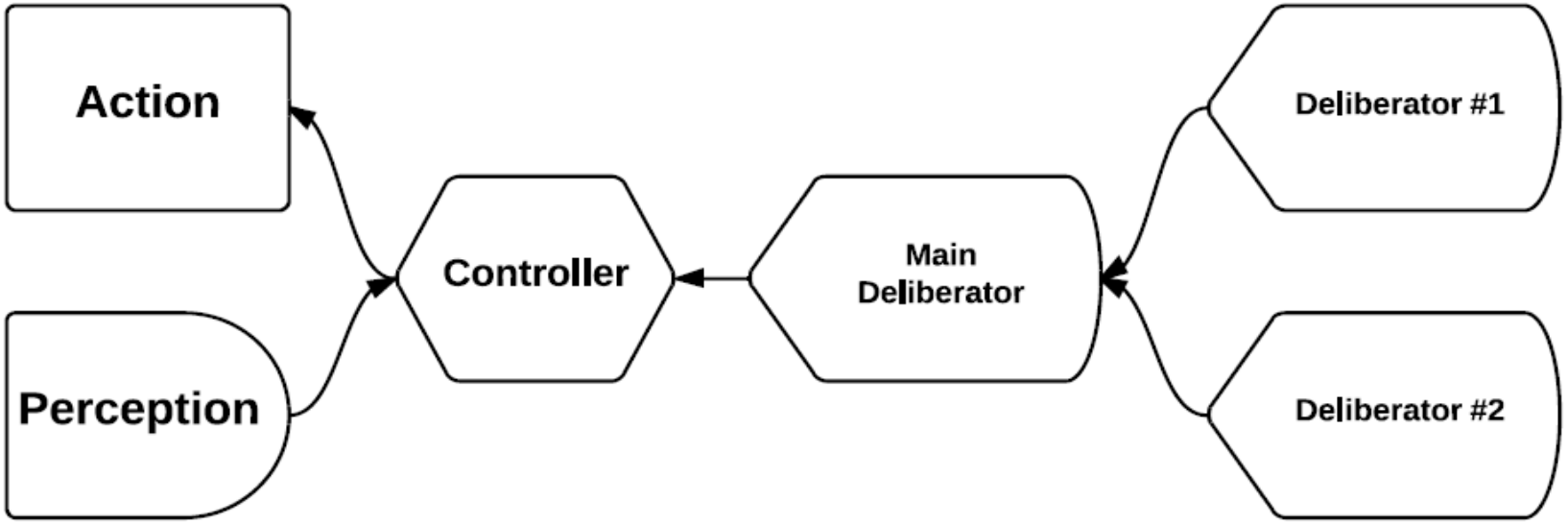
CogBot architecture configurations

- The proposed CogBot architecture
  - Four **basic primitive components** with minimal interface
  - Many architecture instances as **complex configurations** of these components
- Three important classes of configurations follow
  - *P*CDA*
  - *PCD*A*
  - *PCDA-K*

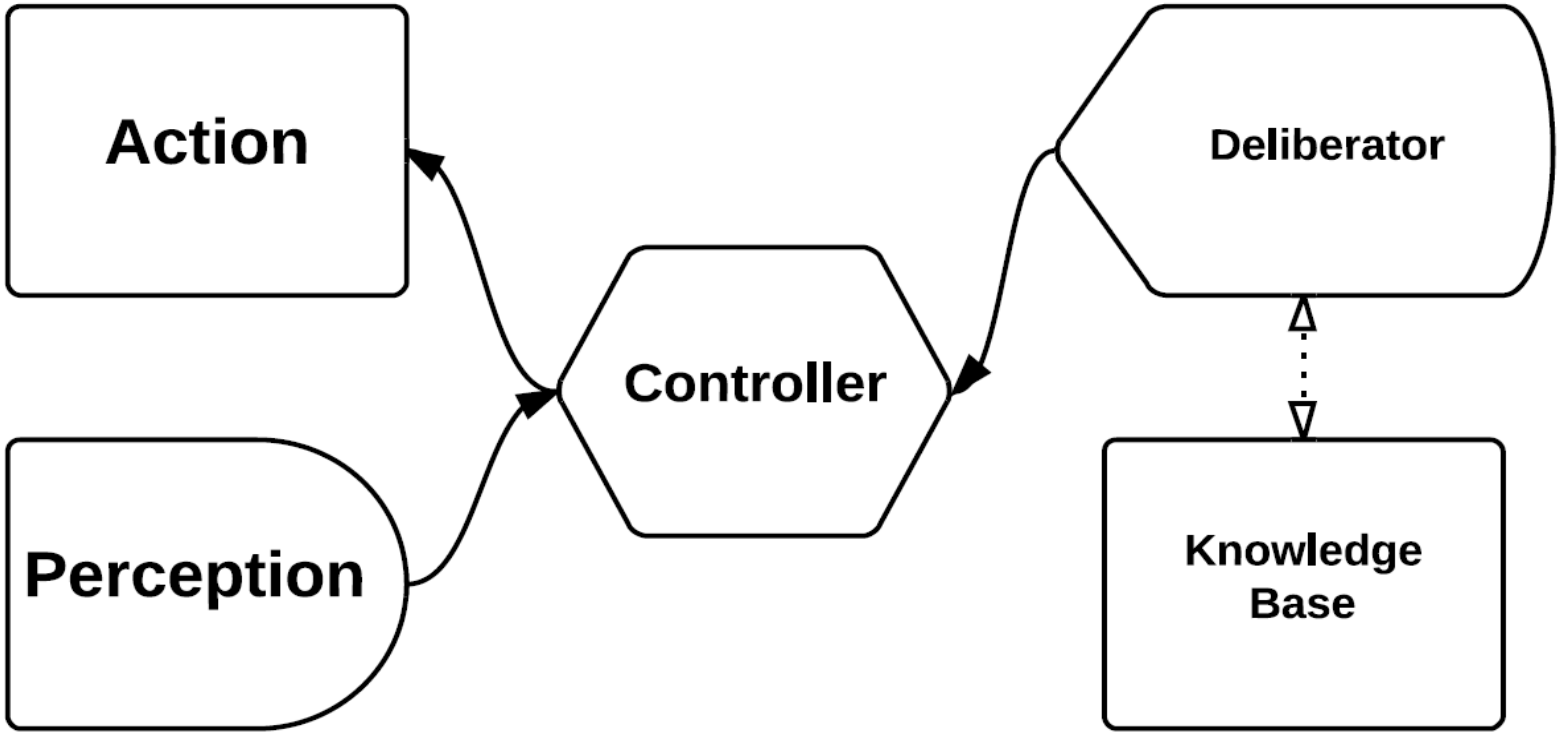
CogBot architecture configurations: *P*CDA*



CogBot architecture configurations: *PCD*A*



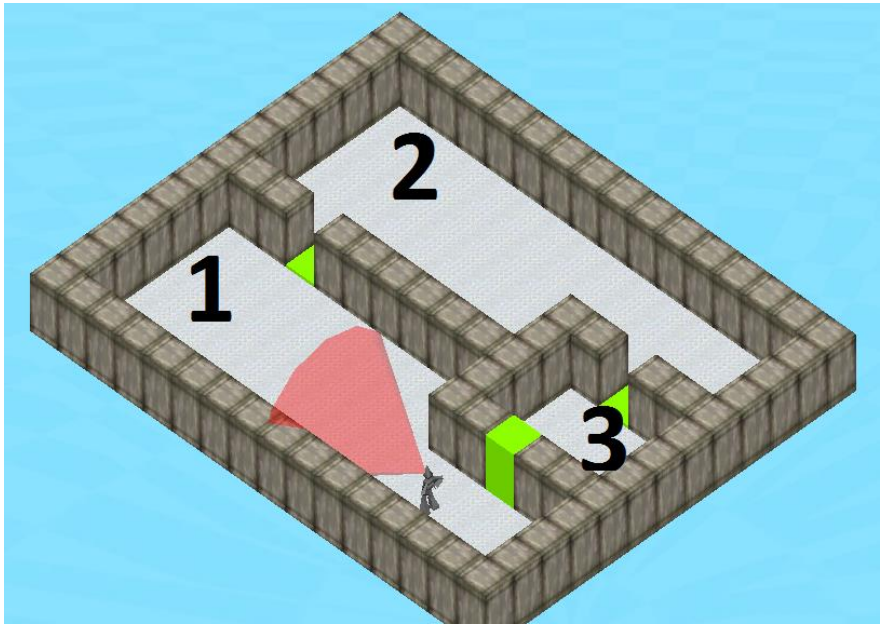
CogBot architecture configurations: *PCDA-K*



Motivating example: Personalized pathfinding

- CogBot *PCDA-K* configuration for the example

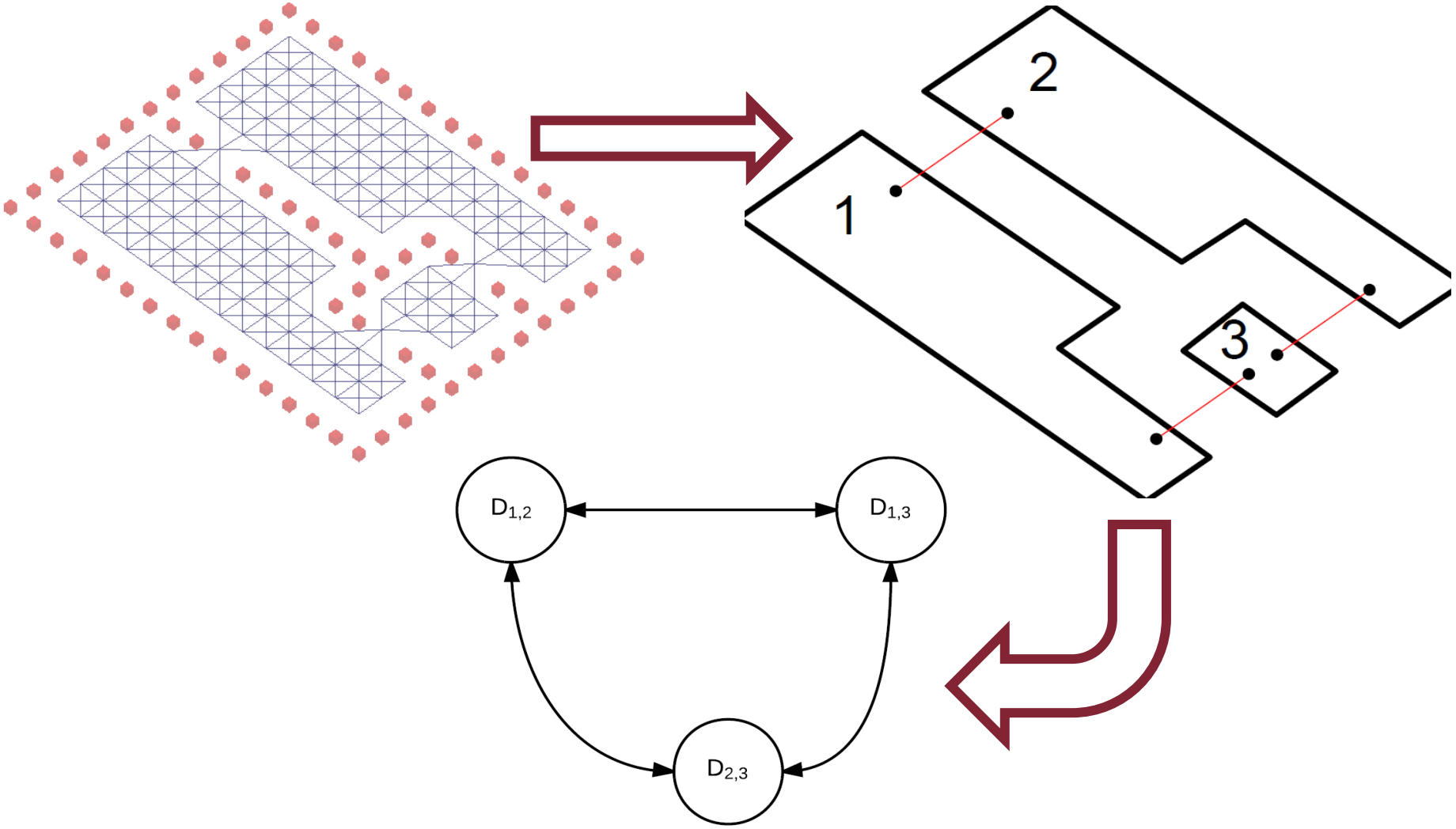
- NPC navigates according to its knowledge of the map connectivity



Motivating example: Personalized pathfinding

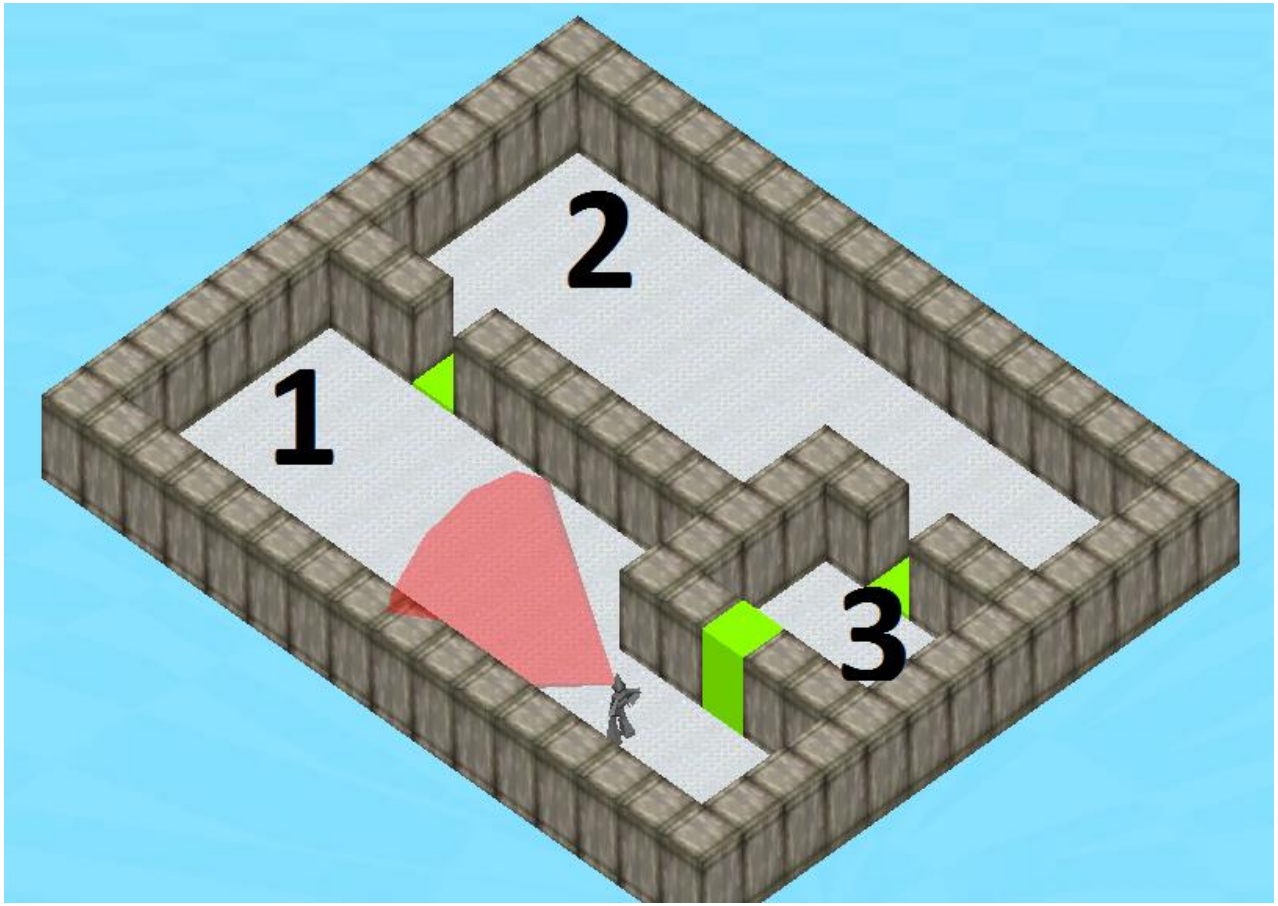
- Scene topology is divided into two different levels:
  - **Low-Level Grid**, shared between each NPC and used by the CogBotAction component.
  - **High-Level Connectivity**, stored inside each NPC's as a CogBotDeliberation component and used to decide a high-level navigation plan.

Motivating example: Personalized pathfinding



Motivating example: Personalized pathfinding

- Demo!



Contributions

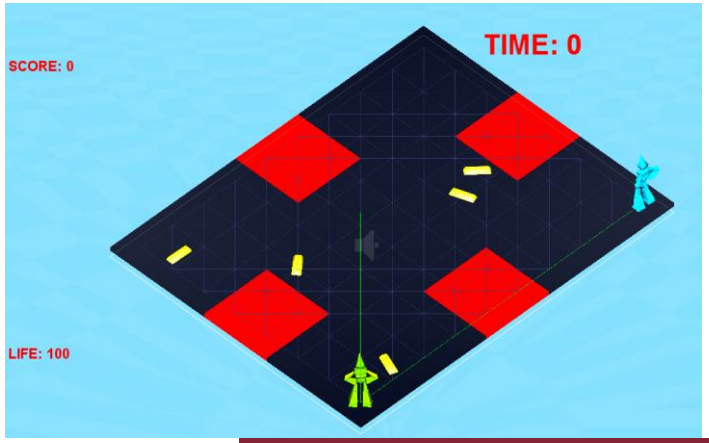
1. Designed a modular NPC architecture according to the objectives and four primitive components
2. Implemented the components and the CDA, PCDA, P*CDA, PCDA-K and PCD*A configurations in the popular Unity3D game engine environment
3. Provide a first step toward a platform for pluggable and hybrid decision-making methods
4. Provide a package with documentation at https://github.com/THeK3nger/unity-cogbot

Contributions

5. We provide GridWorld, a test-bed for NPCs as intelligent agents in Unity3D:
   - Procedural generation of game-world from text map files
   - Low-level pathfinding cababilities
   - High-level area decomposition capabilities
6. We demonstrate the use of CogBot and GridWorld for NPCs in the personalized pathfinding example
7. We demostrate their use as a prototyping platform for visualizing agent execution and interaction in a virtual and a mixed reality setting.

Contributions

Pac-HuMan mixed-reality game.



Visualization for a SOC system

Future work

- CogBot architecture provides the ground for a number of directions we want to explore
  - Compare different existing decision making methods by abstracting them as different deliberation components in the same game-setting.
  - Build NPCs that are able to dynamically change the underlying deliberation method depending on the state of the game (e.g., the vicinity of the player).
  - Build NPCs that are able to use a hybrid combination of existing deliberation approaches.