Faculty of Mathematics and Physics
Charles University in Prague
10th March 2014

UT2004 bots made easy!

Pogamut 3

Lecture 2 – Gentle introduction

Warm up

- Fill the short test for this lessons
  - Find the test here (no-ads):
    - http://goo.gl/0h2lwJ
  - Permanent link:
    - https://docs.google.com/forms/d/1uu3X85_pBkeq9PbnhveKzeFwua_VXnPSSdbDqPI4KrQ/viewform
  - 5 minutes limit

Intelligent Virtual Agents
What?

- Software agent *(by Michael Wooldridge)*
  - Embodied intelligent autonomous entity

Intelligent Virtual Agents

What?

- Software agent *(by Michael Wooldridge)*
  - **Embodied** intelligent autonomous entity
    - Body that is subject to some (physical) laws within its environment

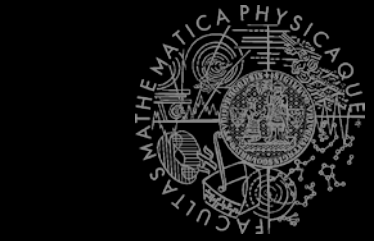
Intelligent Virtual Agents

What?

- Software agent *(by Michael Wooldridge)*
  - Embodied intelligent **autonomous** entity
    - Operating on an owner's behalf but without any interference of that ownership entity

Intelligent Virtual Agents

What?



- Software agent *(by Michael Wooldridge)*
  - Embodied **intelligent** autonomous entity
    - Reactive
    - Proactive
  - *Thermostat may be an agent too!*

Intelligent Virtual Agents

What?

- Software agent *(by Michael Wooldridge)*
  - Embodied **intelligent** autonomous entity
    - Reactive
    - Proactive
    - Social
  - *Okey… 'more' thermostats…*

Intelligent Virtual Agents
What?

- Software agent *(by Michael Wooldridge)*
  - Embodied intelligent autonomous entity
    - Reactive
    - Proactive
    - Social
- Intelligent Virtual Agent *(IVA)*
  - **Specific** software agent **type**
    - Wholly and movably embodied within Complex virtual environment / world
    - Acts under bounded rationality

Intelligent Virtual Agents
What?

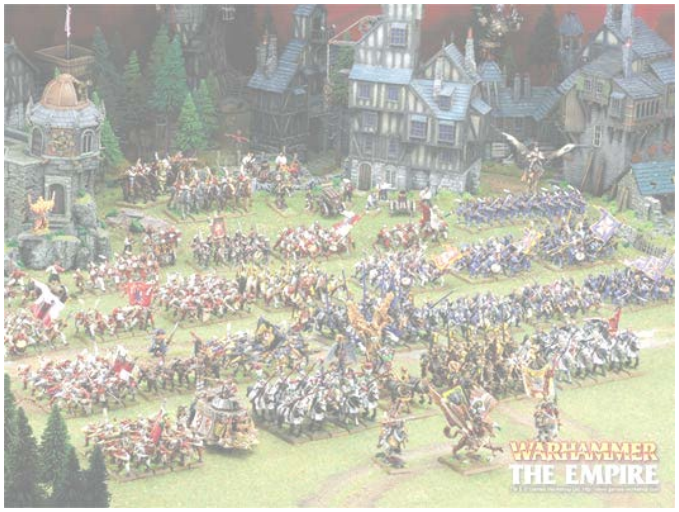
- Software agent *(by Michael Wooldridge)*
  - Embodied intelligent autonomous entity
    - Reactive
    - Proactive
    - Social
- Intelligent Virtual Agent *(IVA)*
  - Specific software agent type
    - Wholly and movably embodied within **Complex virtual environment (… ?)**
    - Acts under bounded rationality

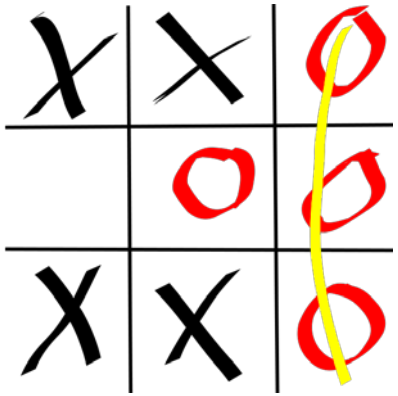
Env. Classification
What can be said?

- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

TicTacToe
What can be said?

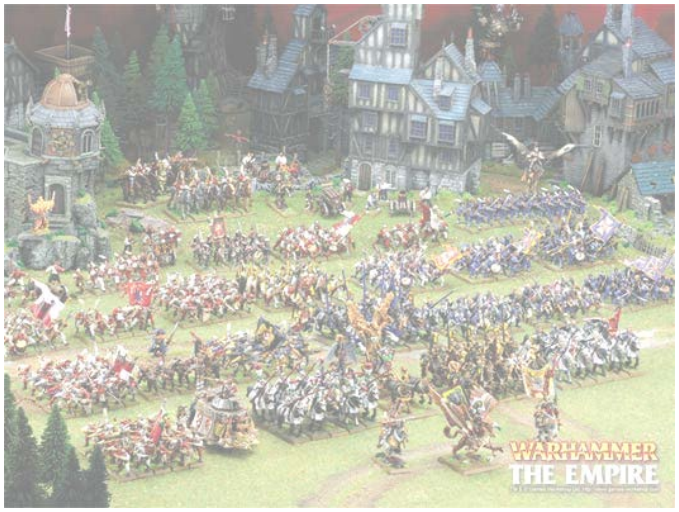
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

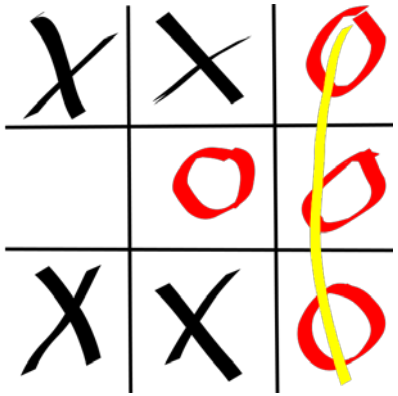




TicTacToe
What can be said?

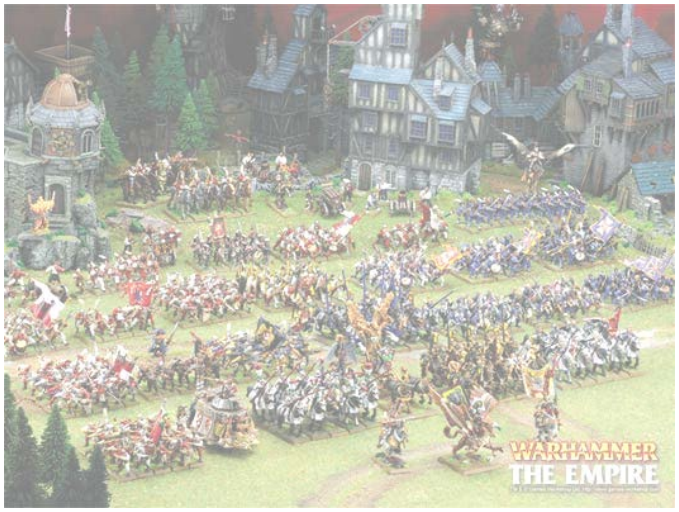
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

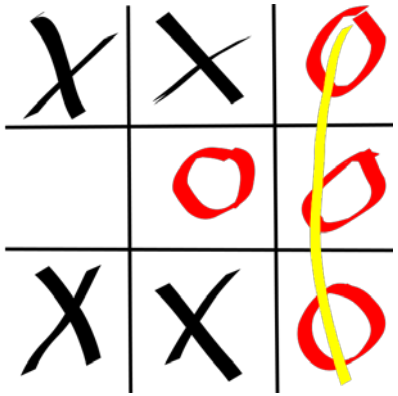




TicTacToe
What can be said?

- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

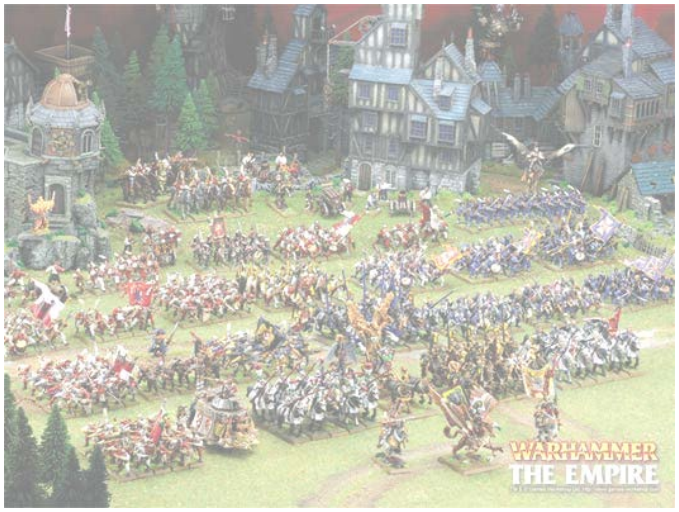


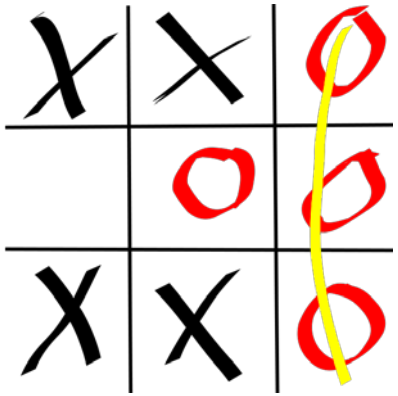


TicTacToe

What can be said?

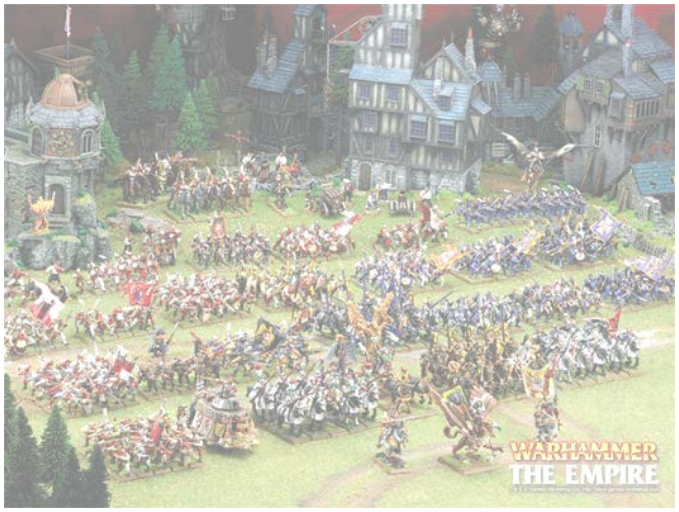
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

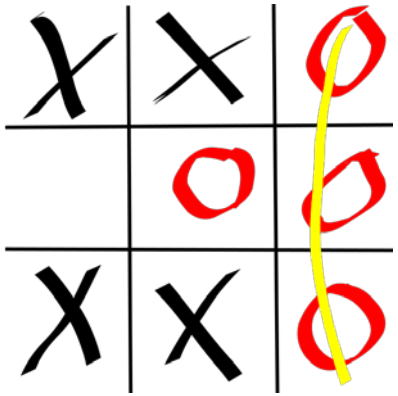




TicTacToe
What can be said?

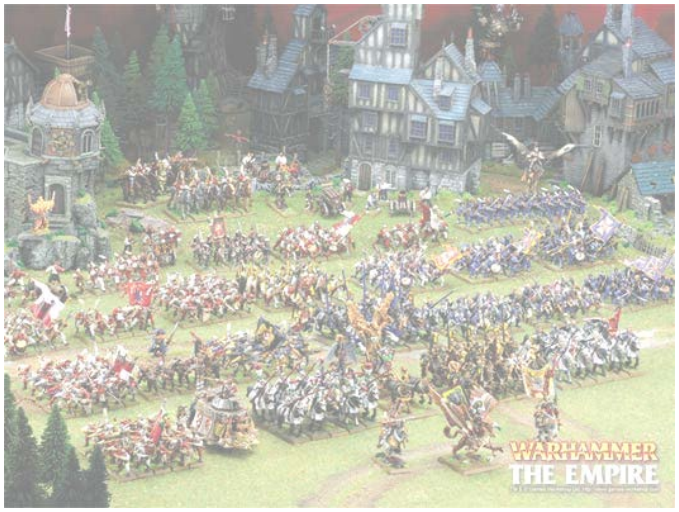
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

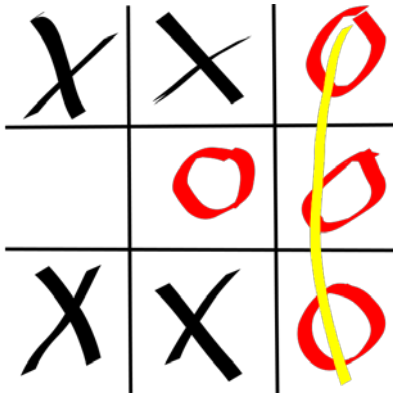




TicTacToe
What can be said?

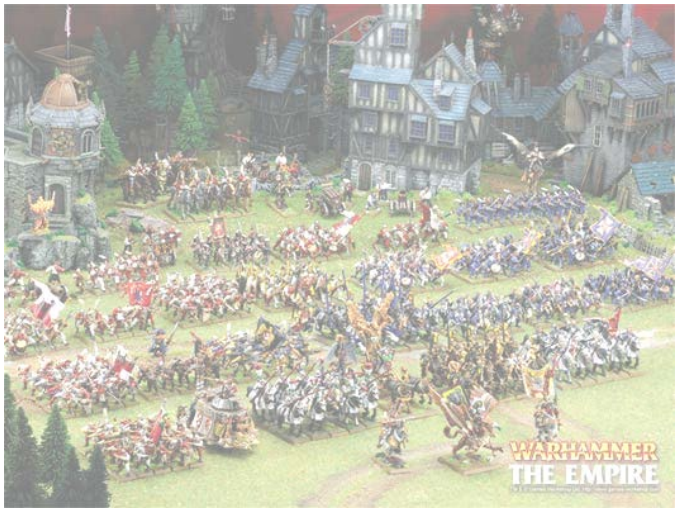
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

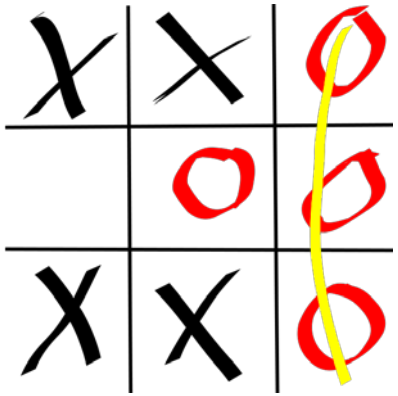




TicTacToe
What can be said?

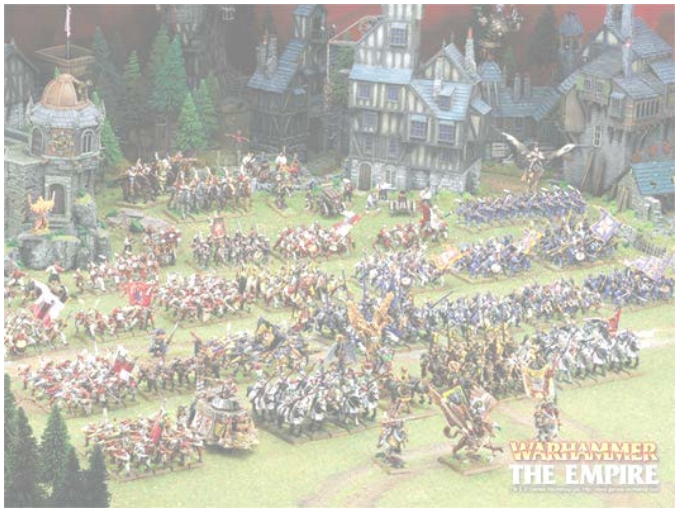
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

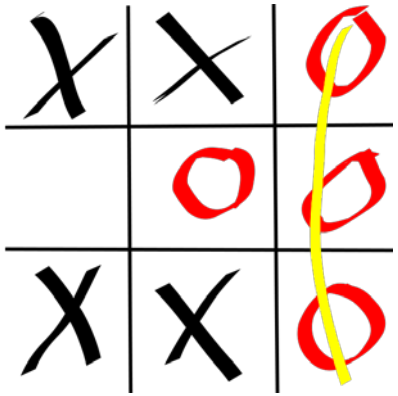




TicTacToe
What can be said?

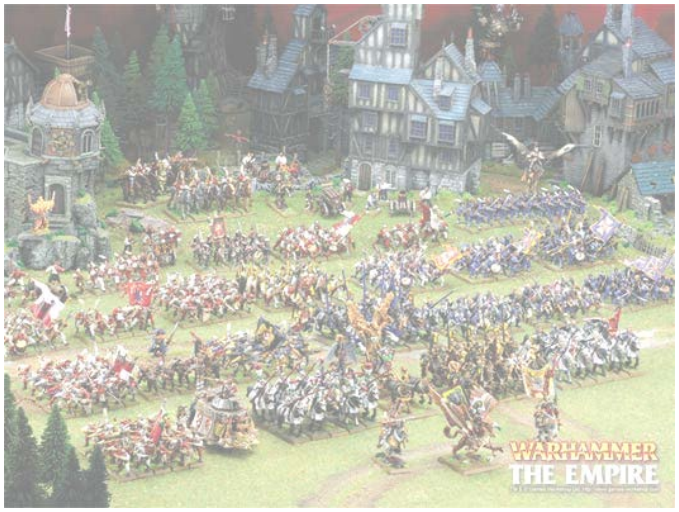
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

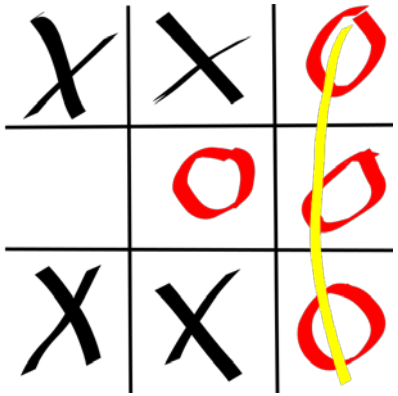




TicTacToe
What can be said?

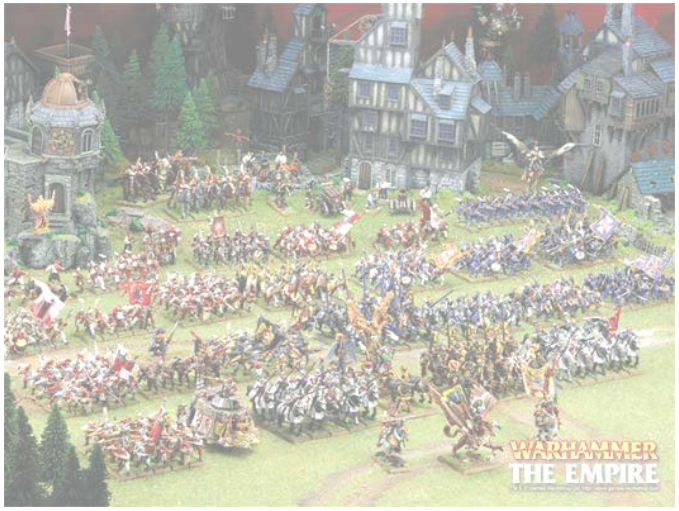
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

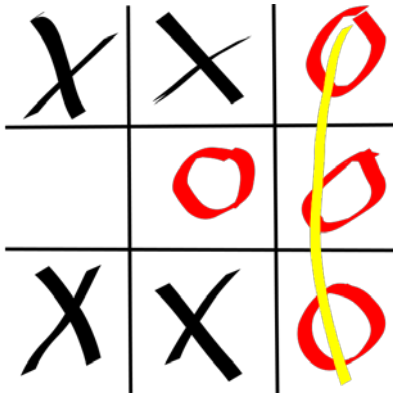




TicTacToe
What can be said?

- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

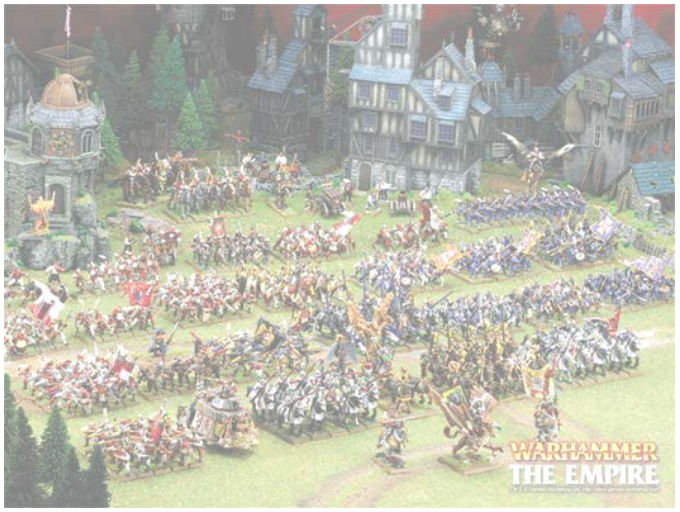


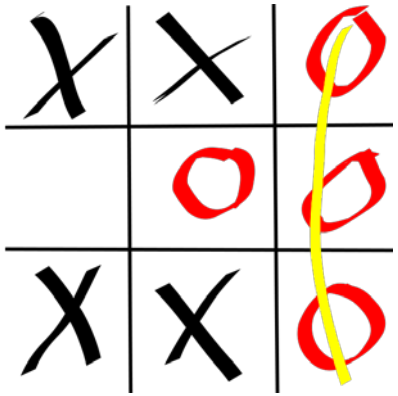


TicTacToe

What does it mean?

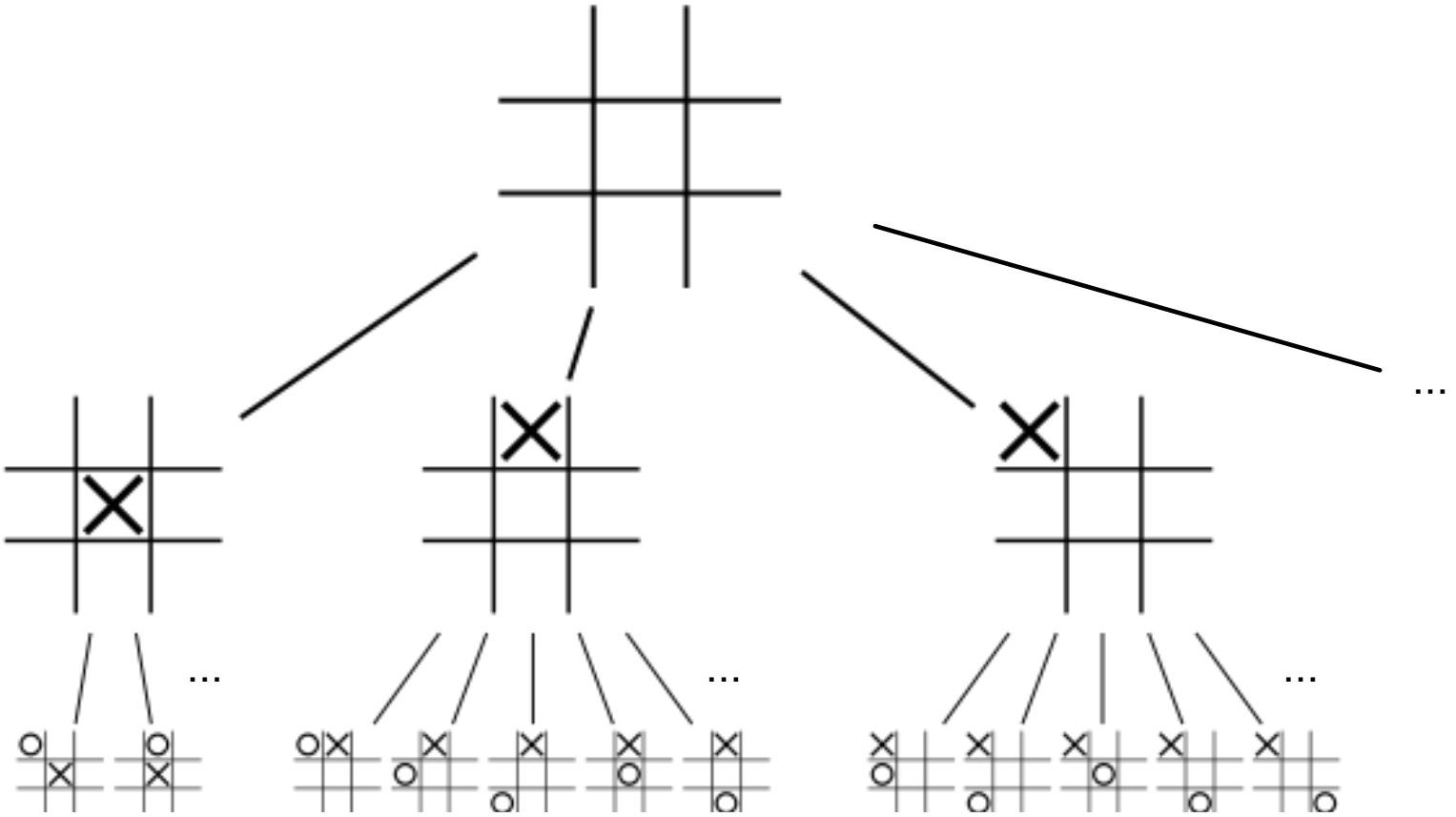
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy





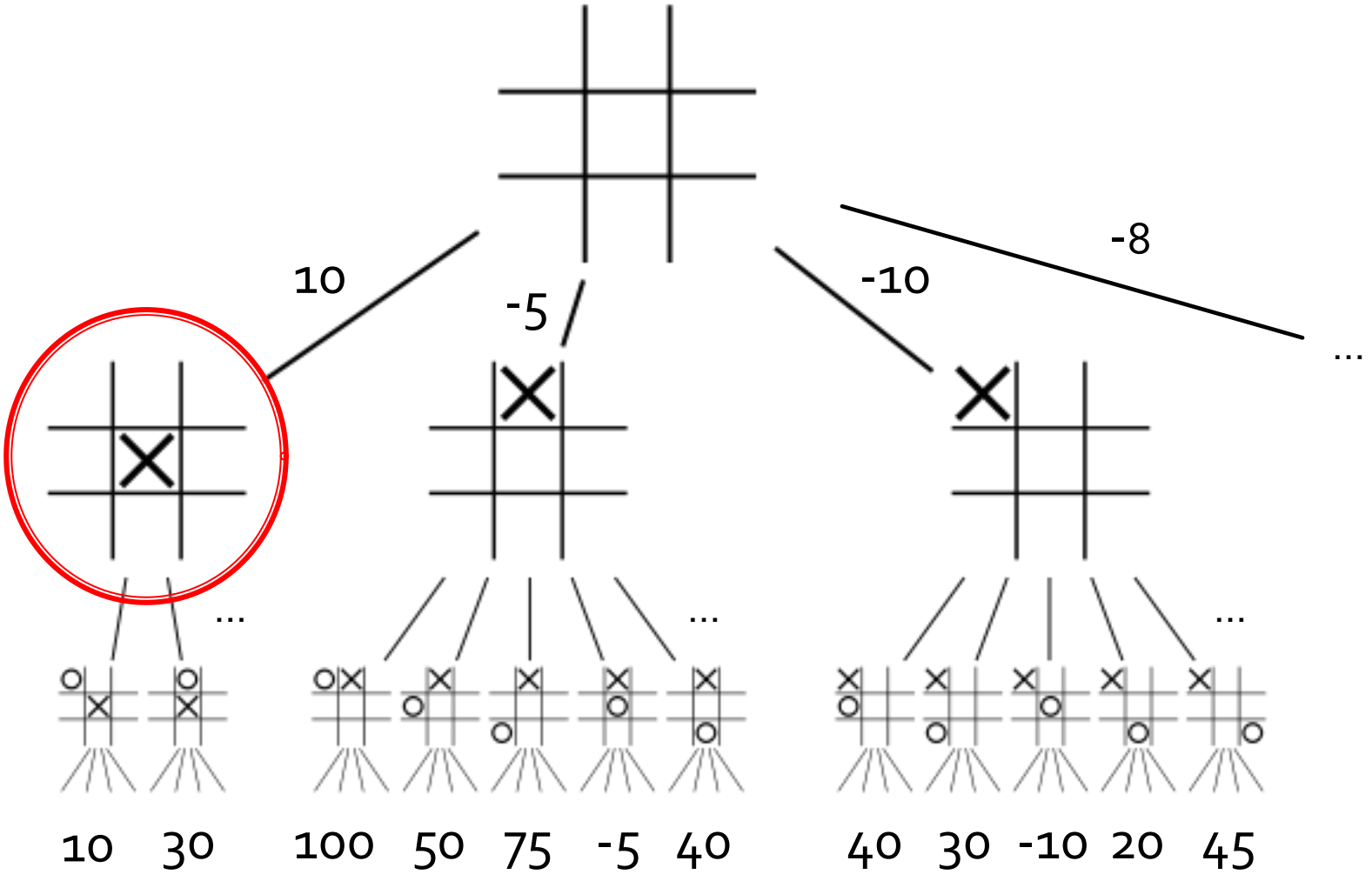
„Reasoning as search“

-- Alan Newell



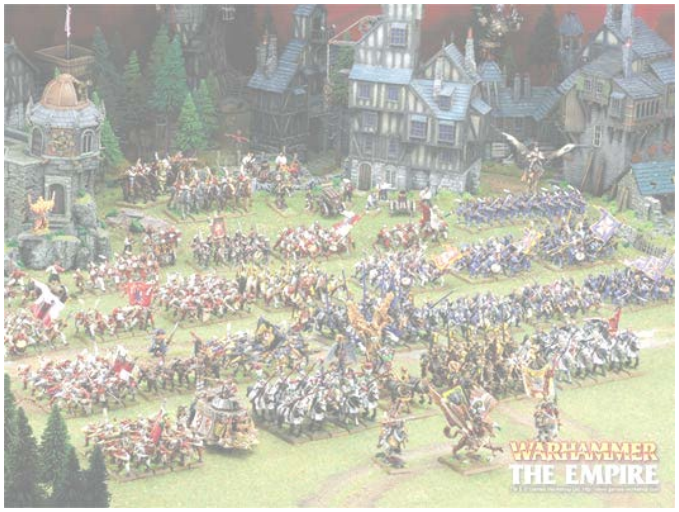
„Reasoning as search“

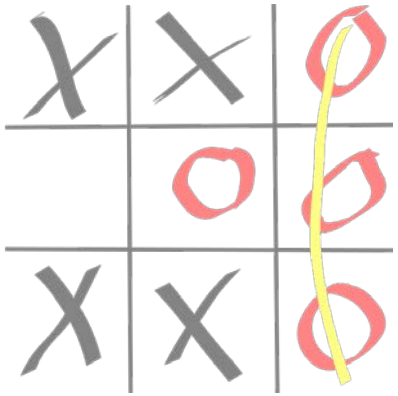
=> MIN-MAX algorithm + modifications



Env. of UT2004?

- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic
- Discrete *vs.* Continuous
- Known *vs.* Unknown
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy

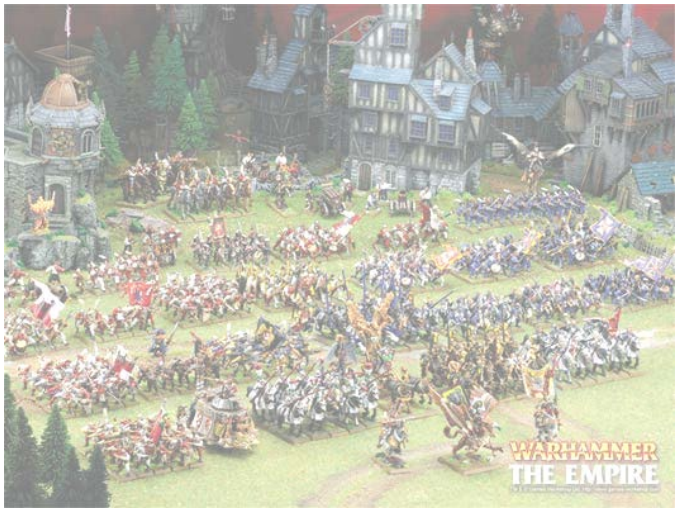


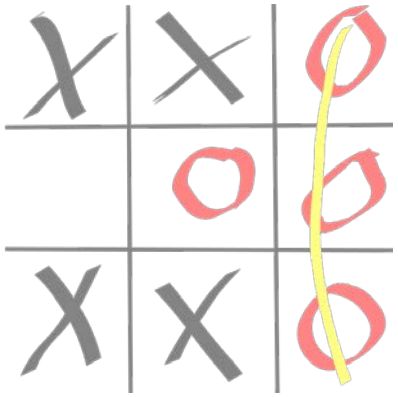


Env. of UT2004

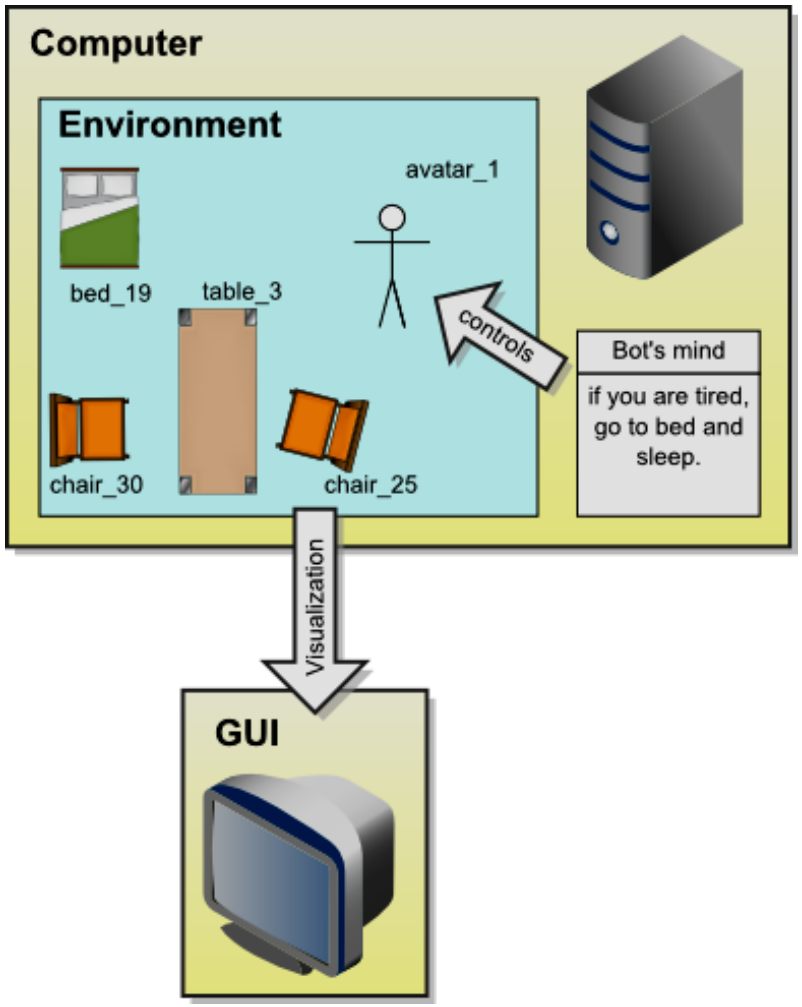
The (almost) worst case imaginable!

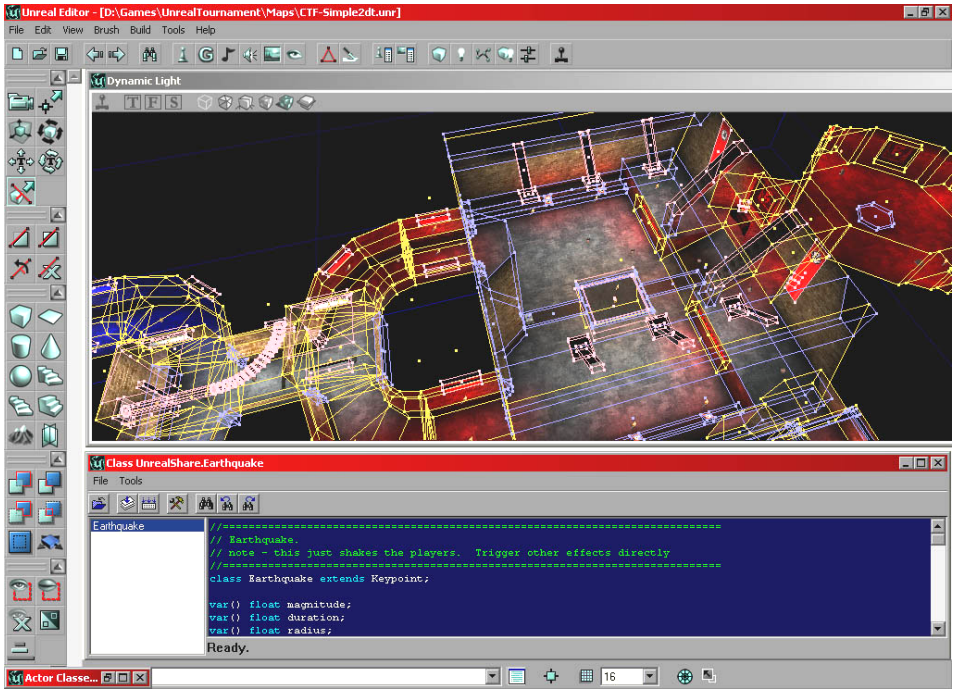
- Fully *vs.* Partially observable
- Episodic *vs.* Sequential
- Static *vs.* Dynamic
- Single *vs.* Multi agent
- Deterministic *vs.* Stochastic (weakly)
- Discrete *vs.* Continuous
- Known *vs.* Unknown (weakly)
- Turn-based *vs.* Real-time
- Noiseless *vs.* Noisy





Virtual worlds





IVAs and Virtual worlds

Environment state (E)

Memory (S)

Perception (P)

Action (A)

1. Part of environment state E is exported to the agent p(E) = P

2. Agent performs action-selection: f(P,S) -> AxS

3. Actions are carried out in the environment: a(A^n,E) -> E

IVAs and Virtual worlds

Environment state (E)

Dynamic world

Non-complete information

Perception (P)

Action (A)

Actions mail fail!

Memory (S)

Inaccurate

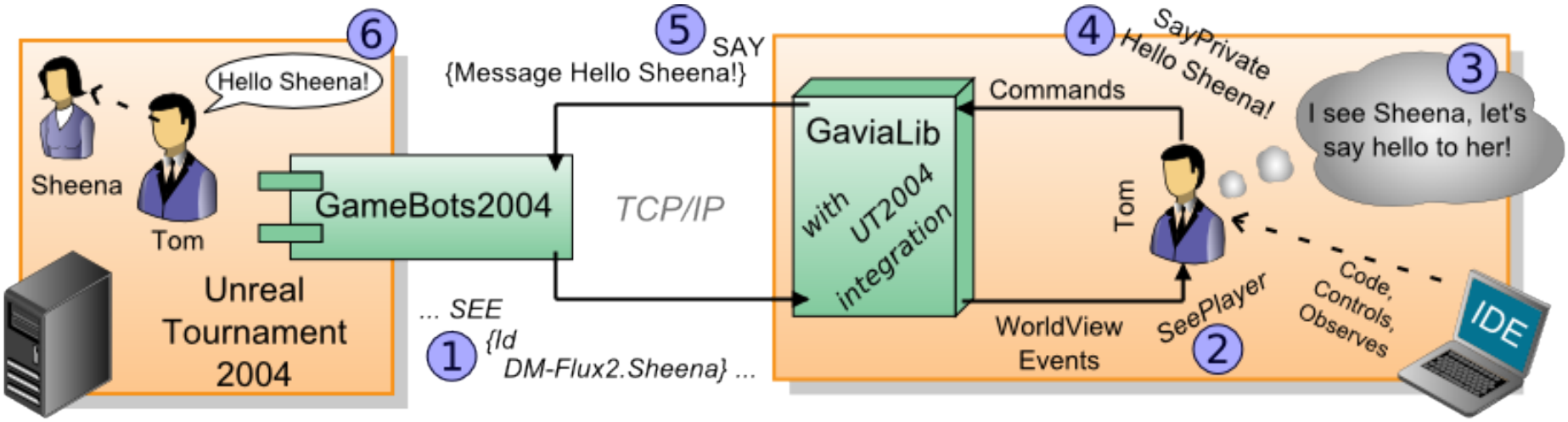
1. Part of environment state E is exported to the agent p(E) = P

2. Agent performs action-selection: f(P,S) -> AxS

3. Actions are carried out in the environment: $a(A^n,E) \rightarrow E$

Pogamut 3 platform

UT2004 and IVAs



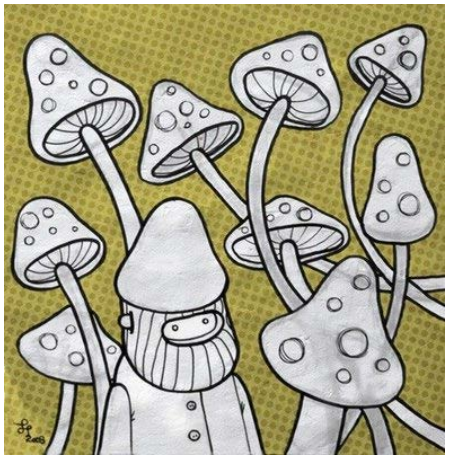
UT2004 is providing action execution function *a*.

GameBots2004 mediates decisions to UT2004 and implements partial observability function *p*.

Pogamut 3 provides observe function *o*.

You have to supply reason function *r*, decide function *d* and possibly extra memory states S.

Decision Making Systems



- Reactive DMS
- Mushroompicker Cyril

Initial state: not_at_home AND picking_mushrooms

1. **IF** in_front_of_obstacle **THEN** change_rotation
2. **IF** full_basket **AND** picking **THEN** stop_picking
3. **IF** see_mushroom **AND** picking **THEN** put_it_to_basket
4. **IF** noon **AND** picking **THEN** stop_picking
5. **IF** at_home **THEN** end
6. **IF** picking **THEN** random_walk
7. **IF** not_picking **THEN** go_home

Pogamut Interface
World / Agent

- **WorldView**
  - A sort of working memory storing all the information bot knows about environment
  - Or a bot current overview of the world
  - Access by **this.world** or **this.getWorldView()**
- **Act**
  - Interface enabling to send bot commands – move to location, start shooting, jump, etc.
  - Access by **this.act** or **this.getAct()**

Pogamut API
Basics

- **In JavaDoc**
  - http://pogamut.cuni.cz/pogamut_files/latest/doc/javadoc/
- **Bot messages**
  - Provide bot with information about environment
  - All of them are subclasses of **InfoMessage** object
- **Bot commands**
  - Allow bot to do things in environment (move, shoot…)
  - All of them are subclasses of **CommandMessage** object

Pogamut API
Bot messages

- Provide information about environment
- Two types
  - IWorldObject vs. IWorldEvent
- **IWorldObject** – persistent object in the game that is typically located (ILocated) and can be seen (IViewable)
  - Is stored in WorldView
- **IWorldEvent** – marks one event in the environment
  - Is not stored and can be missed
  - Listen to events through listeners

Pogamut web

Main web

- http://pogamut.cuni.cz/

JavaDoc (IMPORTANT!)

- http://pogamut.cuni.cz/pogamut_files/latest/doc/javadoc/

Lecture web

- http://pogamut.cuni.cz/pogamut-devel/doku.php?id=lectures

Tutorials

- http://pogamut.cuni.cz/pogamut_files/latest/doc/tutorials/

Pogamut manual installation Win32

- http://pogamut.cuni.cz/main/tiki-download_file.php?fileId=22

Pogamut on Linux (external)

- http://cicolink.blogspot.com/2011/11/unreal-tournament-2004-create-bot-with.html

Installation of Pogamut

Step 1: Install Pogamut

- Run Pogamut installer found in Download section at
  - **http://pogamut.cuni.cz**

Import bot project

Step 2: Create new bot project

- Follow the tutorial at:
  - **http://pogamut.cuni.cz/pogamut_files/latest/doc/tutorials/OpeningExamples.html**

Tutorial 1 – Empty bot

- Get the bot from our lecture site
- We look into the basics of Pogamut bot methods and API…
- See the tutorial:
  - http://pogamut.cuni.cz/pogamut_files/latest/doc/tutorials/EmptyBotTutorial.html

…

- Let's fool around ☺

Starting Pogamut Bot

1. Starting the game environment
   - UT2004 dedicated server
   - Start->Programs->Vyvojove Nastroje->Pogamut->run GameBots DM server
2. Starting the vizualizator (the game UT2004)
   - Start->Programs->Vyvojove Nastroje->Pogamut->run UT2004
3. Starting the bot itself
   - Inside NetBeans – right click the project and select Run

Tutorial 2 – Simple bot

- Listeners – listening to changes in the environment
- See the tutorial:
  - http://pogamut.cuni.cz/pogamut_files/latest/doc/tutorials/ResponsiveBotTutorial.html

…

- Let's fool around again!

Assignment (or HomeWork)

- Extend EmptyBot:
  1. To listen to the player commands
     - If I say "hi", bot responds
     - "Start following" – bot starts following
     - "Stop following" – bot stops following
  2. Remember last position of the player and if the player is lost, run to that location
  3. If the bot doesn't see the player, start turning around to scan your surroundings

Assignment (CheatSheat)

- Listen to **GlobalChat** event to receive text messages
- Use **SendMessage** command to send text messages to the game
- Module **this.players** holds information about other players in the game
- Module **this.move** provides basic locomotion commands
- You can communicate with your bot from within UT2004 by pressing TAB and typing 'say hi' (without quotes)

Send your assignments to

- Completely zip-up your project(s) folder
  - **WITHOUT** the **target** folder!

- Send it to:
  - Michal Bída (Monday practice lessons)
    - michal.bida@gmail.com
  - Jakub Gemrot (Tuesday practice lessons)
    - jakub.gemrot@gmail.com