OPERAČNÍ PROGRAM PRAHA ADAPTABILITA

EVROPSKÝ SOCIÁLNÍ FOND

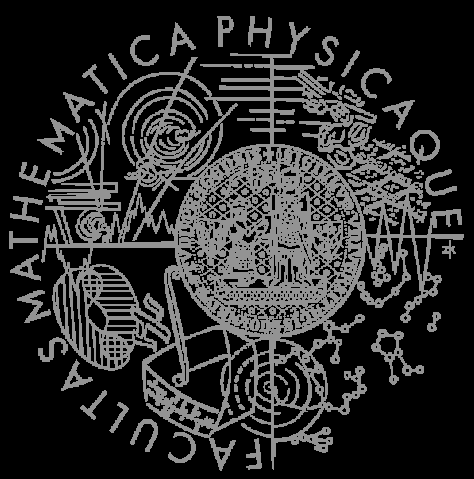
EVROPSKÁ UNIE

Pogamut 3
Lekce 3 – Běhání kolem

PRAHA & EU

INVESTUJEME DO VAŠÍ BUDOUCNOSTI

Faculty of Mathematics and Physics
Charles University in Prague
11th March 2013



UT2004 bots made easy!

Pogamut 3

Lecture 3 – Running Around

Tag! Tournament

Warm Up!

- Fill the short test for this lessons
  - 5 minutes limit
  - http://alturl.com/of3kp

Assignment 2 Revisited

Console/FollowBot

```
private UnrealId followTarget = null;

@EventListener(eventClass = GlobalChat.class)
protected void handleChat(GlobalChat event) {
    if (event.getText().contains("hi"))
      body.getCommunication()
          .sendGlobalTextMessage("Hi");
    if (event.getText().contains("start follow")) {
      followTarget = event.getId();
    }
    if (event.getText().contains("stop follow"))
      followTarget = null;
}
```

```
public void logic() throws PogamutException {
  if (followTarget != null) {
    Player followPlayer = players
                          .getPlayer(followTarget);
    if (info.atLocation(followPlayer.getLocation()) &&
          !followPlayer.isVisible()) {
      move.turnHorizontal(30);
    } else {
      move.moveTo(followPlayer);
    }
  }
}
```

Assignment 2 Revisited

Console/FollowBot

```
private Boolean following = false;
private Boolean jumping = false;
private Boolean searching = false;
private Location search_location;
private Location last_location;

@EventListener(eventClass = GlobalChat.class)
protected void handleChat(GlobalChat event) {
    if (event.getText().contains("hi"))
      body.getCommunication()
          .sendGlobalTextMessage("Hey you");
    if (event.getText().contains("follow")) {
      this.following = !this.following;
      this.searching = false;
    }
    if (event.getText().contains("jump"))
      this.jumping = !this.jumping;
}
```

```
public void logic() throws PogamutException {
  if (this.following) {
    if (this.players.canSeePlayers()) {
      Player pl =
        this.players.getNearestVisiblePlayer();
      this.search_location = pl.getLocation();
      this.searching = true;
      this.move.moveTo(pl);
    } else {
      if (searching) {
        this.move.moveTo(this.search_location);
        if (this.getInfo()
            .atLocation(this.search_location))
          this.searching = false;
      } else
        this.move.turnHorizontal(30);
    }
  }
  if (this.jumping) act.act(new Jump());
}
```

Motivation

>>> Why am I sitting here?

<<< We're going to dive into PogamutUT2004 platform … technically.

>>> Great, just another proprietary library…

<<< Correct, but:

<<< 1) you have to deal with them everywhere,

<<< 2) platform is created around universal principles, you will learn what to look for in other game engines.

>>> Really… *[skeptical face]*

<<< We can only show you the door, you are the one who has to go through it… ;-)

Today's menu

1. **Big Picture**
2. How to see
   - `Self`, `Player`, `Location`, `Rotation`, `Velocity`
   - `this.info`, `this.players`
3. How to move
   - `Move`, `Jump`, `Dodge`
   - `this.move`
4. Tag! Game
   - Rules, Map
   - `TagMap`
5. How to think
   - Intelligence by design
6. Tag! Tournament Announcement

Big Picture

Environment state (E)

Perception (P)

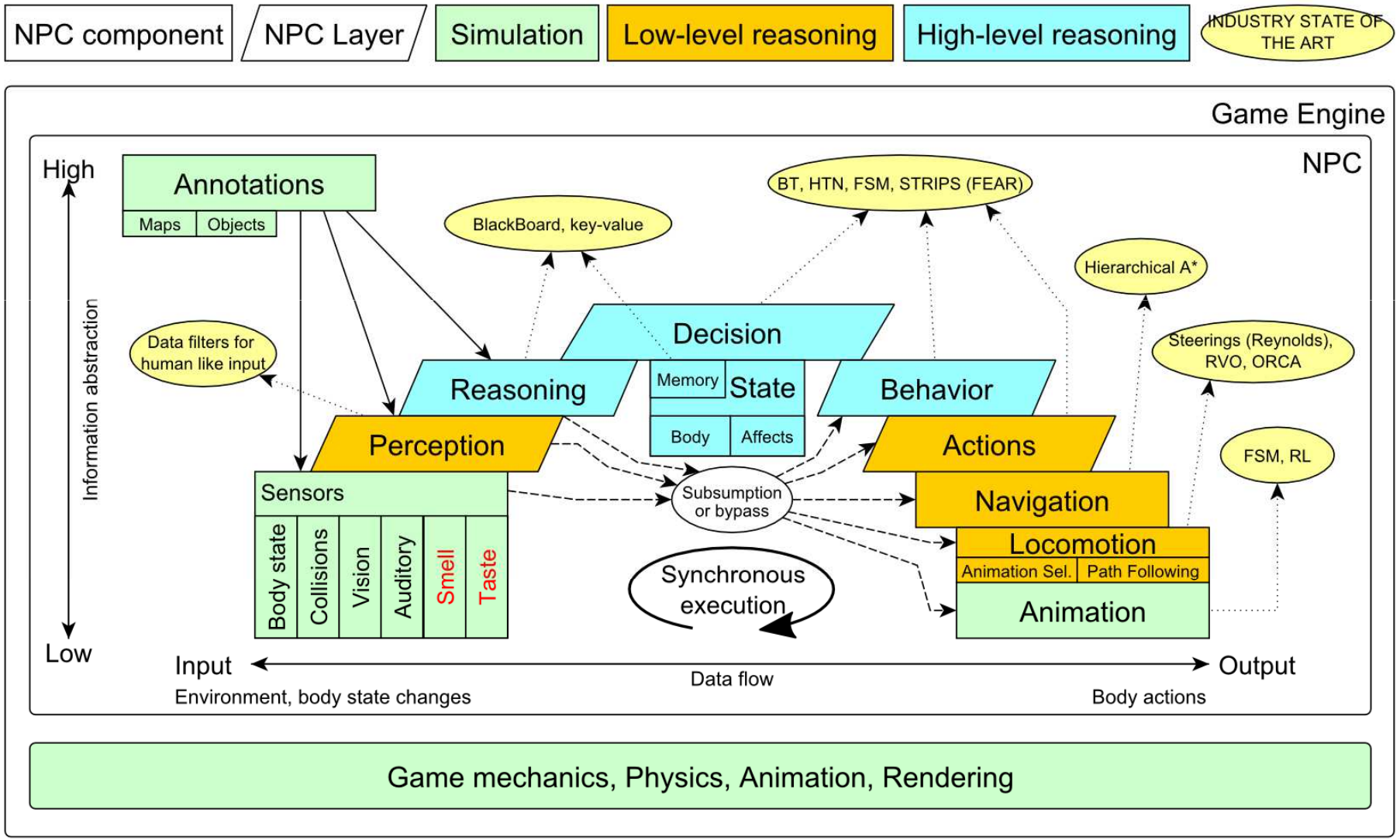
Action (A)

Memory (S)

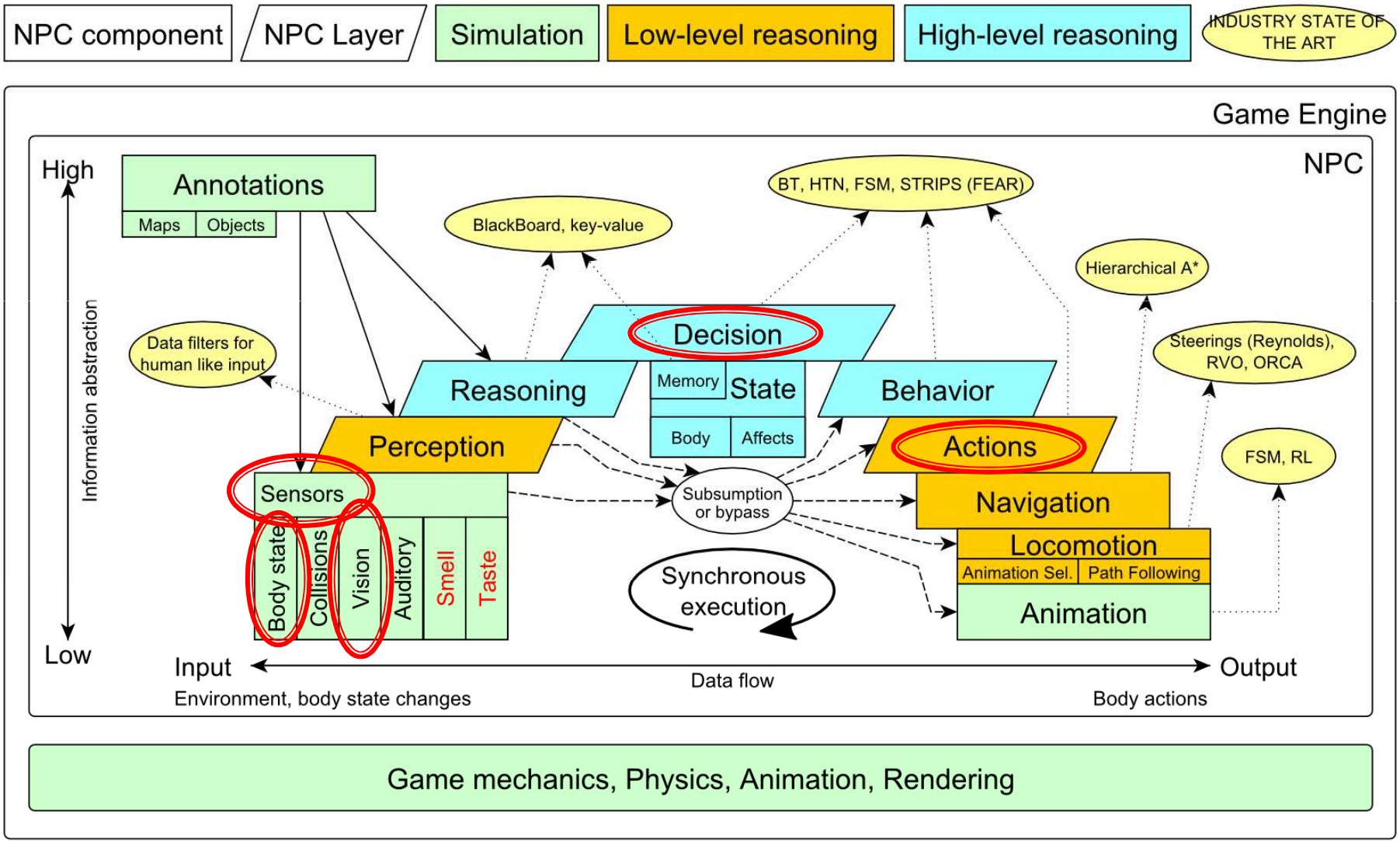
1. Part of environment state E is exported to the agent p(E) = P
2. Agent performs action-selection: f(P,S) -> AxS
3. Actions are carried out in the environment: $a(A^n,E) \rightarrow E$

What if we dive deeper?

Big Picture


NPC component
NPC Layer
Simulation
Low-level reasoning
High-level reasoning
INDUSTRY STATE OF THE ART
Game Engine
NPC
High
Low
Information abstraction
Annotations
Maps
Objects
BlackBoard, key-value
BT, HTN, FSM, STRIPS (FEAR)
Hierarchical A*
Steerings (Reynolds), RVO, ORCA
FSM, RL
Data filters for human like input
Decision
Reasoning
Memory
State
Body
Affects
Behavior
Perception
Actions
Sensors
Subsumption or bypass
Navigation
Locomotion
Animation Sel.
Path Following
Animation
Body state
Collisions
Vision
Auditory
Smell
Taste
Synchronous execution
Input
Environment, body state changes
Data flow
Output
Body actions
Game mechanics, Physics, Animation, Rendering

Big Picture
Today



Today's menu

1. Big Picture
2. **How to see**
   - **`Self`, `Player`, `Location`, `Rotation`, `Velocity`**
   - **`this.info`, `this.players`**
3. How to move
   - `Move, Jump, Dodge`
   - `this.move`
4. Tag! Game
   - Rules, Map
   - `TagMap`
5. How to think
   - Intelligence by design
6. Tag! Tournament Announcement

How to see?

Sensors – Body state, Vision

- `IWorldObjects`
  - `Self`, `Player`, *`Item`*, *`NavPoint`*, …
  - `this.world.getSingle(Self.class)`
  - `this.world.getAll(Player.class)`
  - *`this.world.getAll(Item.class)`*
  - *`this.world.getAll(NavPoint.class)`*
- Agent modules
  - `AgentInfo ~ this.info`
  - `Players ~ this.players`
  - *`Items ~ this.items`*
  - *`NavPoints ~ this.navPoints`*
- `Location`, `Rotation`, `Velocity`

How to see?

Sensors – Body state, Vision

- `IWorldObjects`
  - `Self`, `Player`, *`Item`*, *`NavPoint`*, …
  - All objects have unique `UnrealId`
    - Each unique id has single `UnrealId` instance
  - Each unique object has single instance
    - Agent modules are respecting this, no sneaky clone()s

What does it mean for **`Collection`**s?

=> can be used in `Set<UnrealId>`, `Set<Player>`

=> can be used as key in `Map<UnrealId, ?>`, `Map<Player, ?>` without performance hit

How to see?
Sensors – Body state, Vision

- `IWorldObjects`
  - `Self`, `Player`, *`Item`*, *`NavPoint`*, …
  - All objects have unique `UnrealId`
    - Each unique id has single `UnrealId` instance
  - Each unique object has single instance
    - Agent modules are respecting this, no sneaky clone()s

What does it mean for **object update**s?

=> once obtained instances are auto-updated

=> there is no history

How to see?

Sensors – Body state, Vision

- `IWorldObjects` ~ low-level API
  - `this.world.getSingle(Self.class)`
    - Info about your bot
  - `this.world.getAll(Player.class)`
    - Returns `Map<UnrealId, Player>`
    - All players encountered during the session
  - `this.world.getAllVisible(Player.class)`
    - Returns `Map<UnrealId, Player>`
    - All players currently visible (in bot's FOV)
  - `this.world.getAll/Visible(Item.class)`
  - `this.world.getAll/Visible(NavPoint.class)`
  - …

How to see?

Sensors – Body state, Vision

- Agent modules ~ low-level API façades
  - `AgentInfo ~ this.info ~ Self`
  - `Players ~ this.players ~ Player(s)`
  - *`Items ~ this.items ~ Item(s)`*
  - *`NavPoints ~ this.navPoints ~ NavPoint(s)`*

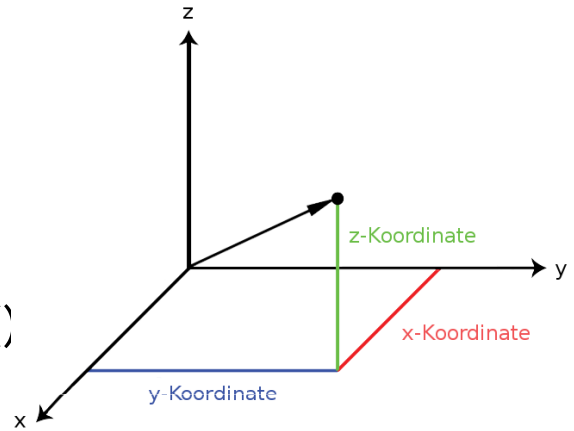
- Advantages:
  1. List of methods with JavaDoc
     => Easier to way to explore Pogamut API
  2. Comprehensibly named methods
     => Easier to read & understand the code

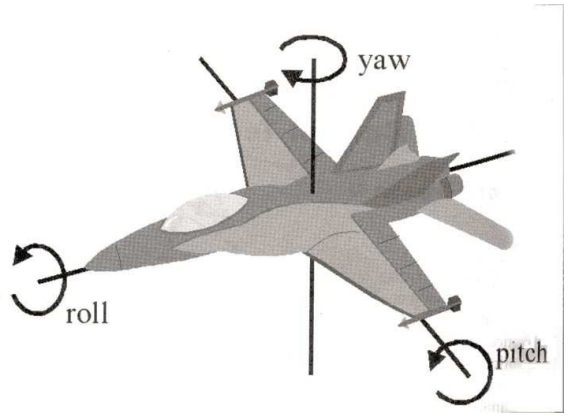
How to see?

Sensors – Body state, Vision

- `Location`
  - X, Y, Z
  - can be used as "vector"
    - add(), sub(), scale(), getDistance(), dot(), cross()
    - rotateXY/XZ/YZ()
- `Rotation`
  - Pitch (XZ), Yaw (XY), Roll (YZ)
- `Velocity`
  - X, Y, Z
- All objects are immutables

  => Can be used in `Set`, `Map`





Today's menu

1. Big Picture
2. How to see
   - `Self`, `Player`, `Location`, `Rotation`, `Velocity`
   - `this.info`, `this.players`
3. **How to move**
   - **`Move, Jump, Dodge`**
   - **`this.move`**
4. Tag! Game
   - Rules, Map
   - `TagMap`
5. How to think
   - Intelligence by design
6. Tag! Tournament Announcement

How to move?
Actions

- `CommandMessages`
  - `Move`, `Jump`, *`Dodge`*
  - `this.act.act(new Move()…)`
  - `this.act.act(new Jump()…)`
  - `this.act.act(new Dodge()…)`

- Agent module
  - `AdvancedLocomotion ~ this.move`

How to move?
Actions

- `CommandMessages` ~ low-level API
  - `Move`
    - You can specify 1 location in advance
    - You can specify focus (where to look while moving), i.e., can be used for strafing
  - `Jump`
    - Can be used for double-jumps as well
  - `Dodge`
    - Can be used for quick direct jump to arbitrary location

How to move?
Actions

- Agent modules ~ low-level API façade
  - `AdvancedLocomotion ~ this.move`
  - All commands wrapped into methods
    - `move.moveTo(), move.strafeTo(), move.jump(), …`
  - Some simple algebra wrapped as well
    - `move.dodgeLeft(), move.dodgeRight(), …`

Today's menu

1. Big Picture
2. How to see
   - `Self`, `Player`, `Location`, `Rotation`, `Velocity`
   - `this.info`, `this.players`
3. How to move
   - `Move`, `Jump`, `Dodge`
   - `this.move`
4. **Tag! Game**
   - **Rules, Map**
   - **`TagMap`**
5. How to think
   - Intelligence by design
6. Tag! Tournament Announcement

Tag! Game
Children play

- Custom "game-mode" for UT2004
- Two roles:
  1. Seeker (having "it")
  2. Runner or Prey
- Seeker has to chase runners to pass „it"
- After passing "it" the *former* seeker is immune to the *new* seeker
- `this.tag` agent module
- Custom map: DM-TagMap
  - Simple rectangle map, no obstacles
  - procedurally decsribed by `TagMap` static methods

Today's menu

1. Big Picture
2. How to see
   - `Self`, `Player`, `Location`, `Rotation`, `Velocity`
   - `this.info`, `this.players`
3. How to move
   - `Move`, `Jump`, `Dodge`
   - `this.move`
4. Tag! Game
   - Rules, Map
   - `TagMap`
5. **How to think**
   - **Intelligence by design**
6. Tag! Tournament Announcement

How to think?
Intelligence by design

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: $a(A^n,E) \rightarrow E$

How to think?
Intelligence by design

Behavior Oriented Design

by Joanna J. Bryson (UK)
http://www.cs.bath.ac.uk/~jjb/web/bod.html

1. Specify top-level decision
   a) Name the behaviors that the bot should do
   b) Identify the list of sensors that is required to perform the behavior
   c) Identify the priorities of behaviors
   d) Identify behavior switching conditions
2. Recursion on respective behaviors until primitive actions reached

Today's menu

1. Big Picture
2. How to see
   - `Self`, `Player`, `Location`, `Rotation`, `Velocity`
   - `this.info`, `this.players`
3. How to move
   - `Move`, `Jump`, `Dodge`
   - `this.move`
4. Tag! Game
   - Rules, Map
   - `TagMap`
5. How to think
   - Intelligence by design
6. **Tag! Tournament Announcement**

Tag! Tournament

Chance to score extra points!

- 4 bots
  - 1 Seeker, 3 Runners (1 of them will be immune…)
- Random groups
- Tournament will be held next week, only bots submitted until Sunday 17.3.2013, 23:59 will participate
- No shooting allowed, no bot speed reconfigurations allowed

Assignment 3

- Download the TagBot project template
- Copy 'map/DM-TagMap.ut2' into UT2004/Maps folder
- Alter UT2004/System/startGamebotsDMServer.bat replacing 'DM-TrainingDay' with 'DM-TagMap'
- Implement both TagBot roles
  - Seeker ~ 5 points
  - Runner ~ 5 points
- Implementations having one role only won't be accepted (~ 0 points)

Send us finished assignment

Via e-mail:

- *Subject*
  - "Pogamut homework 2013 – Assignment X"
    - Replace `'X'` with the assignment number and the subject has to be without quotes of course
    - …or face -2 score penalization
- *To*
  - jakub.gemrot@gmail.com
    - Jakub Gemrot (Monday practice lessons)
  - michal.bida@gmail.com
    - Michal Bída (Thursday practice lessons)
- *Attachment*
  - Completely zip-up your project(s) folder except `'target'` directory and `IDE specific files` (or face -2 score penalization)
- *Body*
  - **Please send us information about how much time it took you to finish the assignment + any comments regarding your implementation struggle**
    - *Information won't be abused/made public*
    - *In fact it helps to make the practice lessons better*
  - Don't forget to mention your full name!

Questions?
I sense a soul in search of answers…

- We do not own the patent of perfection (yet…)
- In case of doubts about the assignment, tournament or hard problems, bugs don't hesitate to contact us!
  - Jakub Gemrot (Monday practice lessons)
    - jakub.gemrot@gmail.com
  - Michal Bída (Thursday practice lessons)
    - michal.bida@gmail.com

OPERAČNÍ PROGRAM PRAHA ADAPTABILITA

EVROPSKÁ UNIE

DĚKUJI ZA POZORNOST