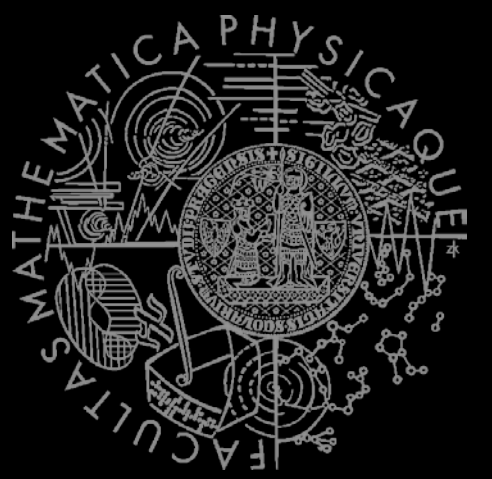
Faculty of Mathematics and Physics
Charles University in Prague
10th March 2015



UT2004 bots made easy!

Pogamut 3

Workshop 3 – Running Around

Tag! Tournament

Assignment 3

Setup

- Start downloading the TagBot project template (~75MB) in advance … now ☺
- Start copying `C:\Program files (x86)\Unreal Anthology\UT2004` into `D:\UT2004`
  - *We will need to modify UT2004 later during the workshop…*

Warm Up!

- Fill the short test for this lessons
  - 7 minutes limit
  - http://goo.gl/PFCTVb

- Permanent link
  - https://docs.google.com/forms/d/1p5Mw5ikEkDXfSIvtl4N94veXA-yooyzw-I7XGYovFYE/viewform

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)

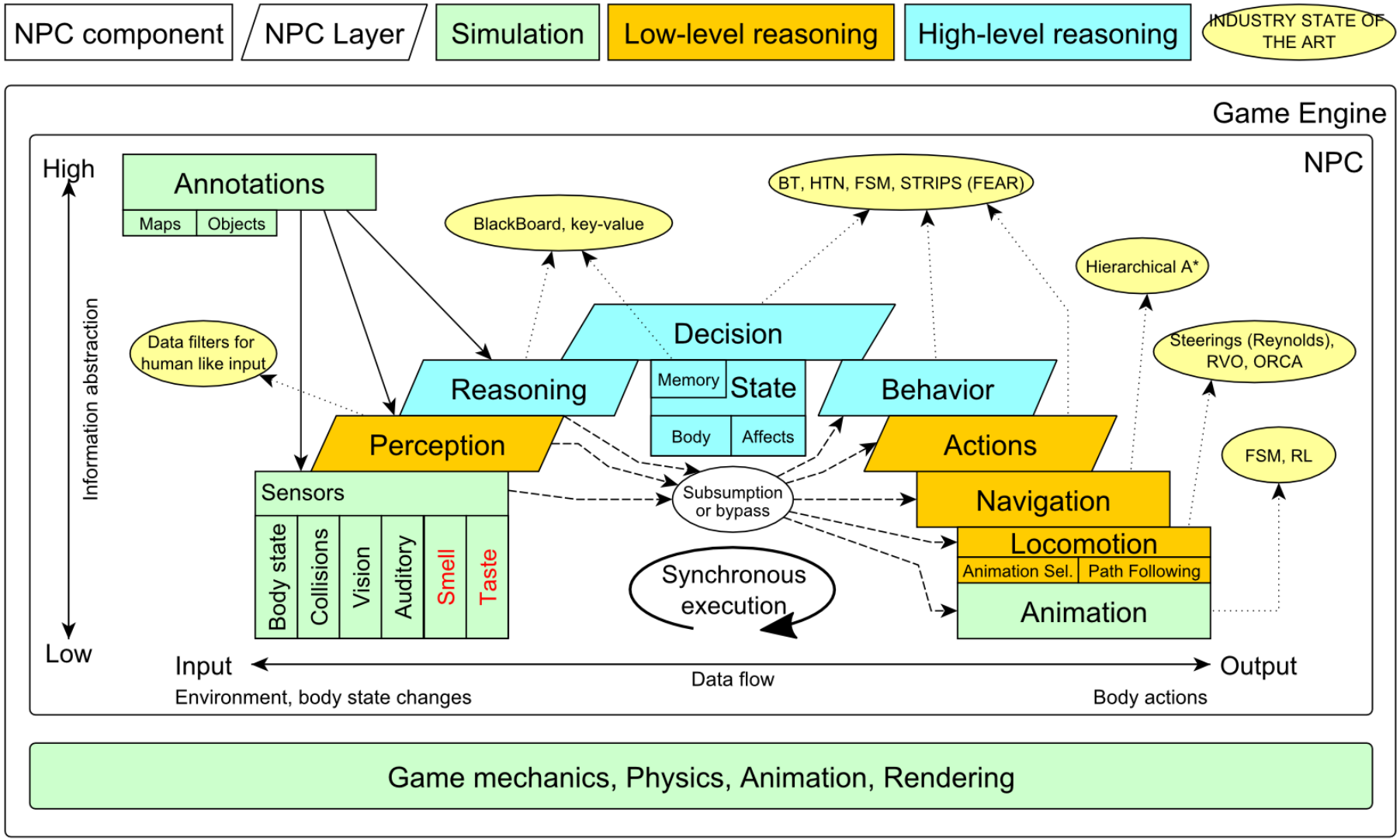
Action (A)

Memory (S)

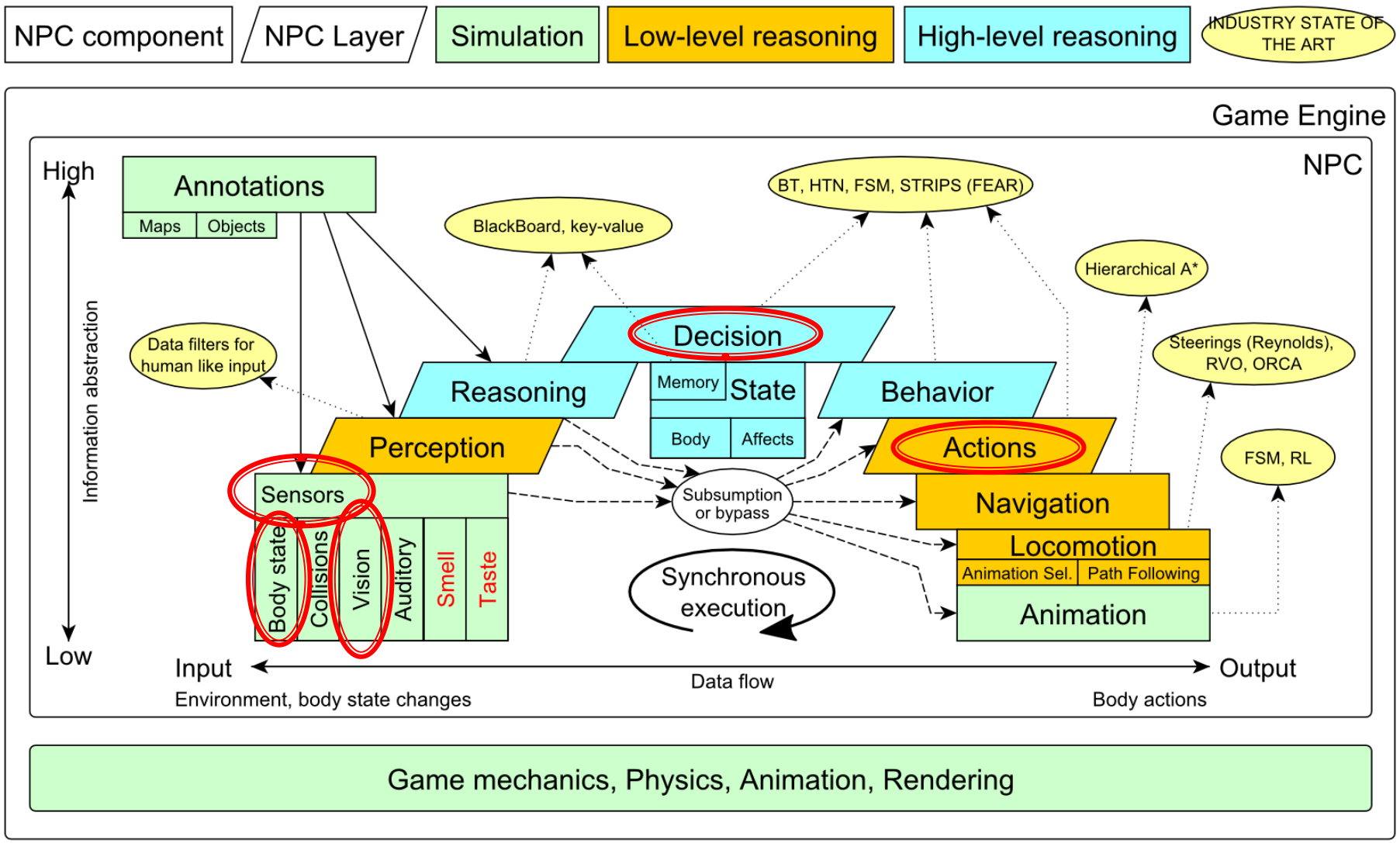
1. Part of environment state E is exported to the agent p(E) = P
2. Agent performs action-selection: f(P,S) -> AxS
3. Actions are carried out in the environment: $a(A^n,E) \rightarrow E$

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*
Data filters for human like input
Decision
Reasoning
Memory
State
Behavior
Steerings (Reynolds), RVO, ORCA
Perception
Body
Affects
Actions
FSM, RL
Sensors
Subsumption or bypass
Navigation
Body state
Collisions
Vision
Auditory
Smell
Taste
Locomotion
Animation Sel.
Path Following
Synchronous execution
Animation
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` (explained later on)

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)`*

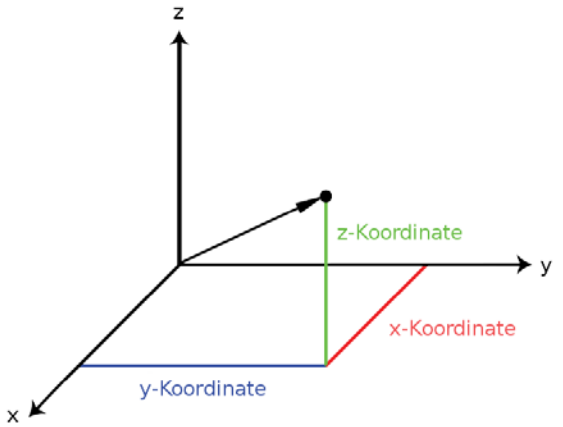
- Advantages:
  1. List of methods with JavaDoc
     => Easier to way to explore Pogamut API
  2. Comprehensibly named methods
     => Easier to read & understand the code

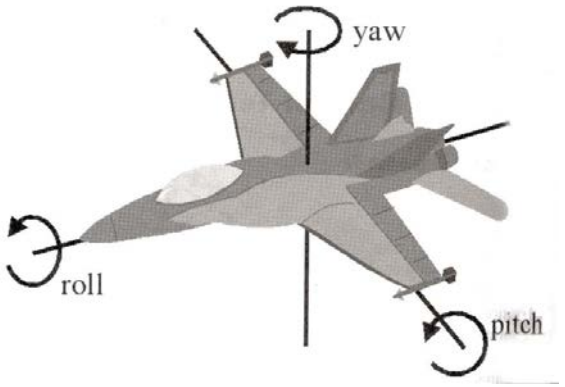
How to see?

Sensors – Body state, Vision

- `Location`
  - X, Y, Z (world space)
  - 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 vector
  - Length is speed in UT units (1UT ~ 1cm)
- 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 Saturday 15.3.2014, 9:00 will participate
- No shooting allowed, no bot speed reconfigurations allowed
- The best 6 bots from Tag! 2014 will participate in the tournament
  - You will have a chance to test your bots against them in advance

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)

Assignment 3

- Note that there are two "main" Java files in the project
- `TagBot`
  - Bot template you have to finish
  - DO NOT ALTER ITS main METHOD!
- `TagGame`
  - Class that starts the match between 4 your bots
  - Use this to test your bot

Assignment 3

Cheat sheet – Strategy – Catcher / Chaser

- Your bot should recognize 3 stages of chasing
  - **Early-stage**
    - You are really far from your target
    - ⇒ You have to quickly shorten this distance
    - ⇒ Use rough double-dodges
  - **Mid-stage**
    - You are trying to corner your target
    - ⇒ Be careful what commands you're issuing, you probably want to avoid "straight" running to your target
  - **Final-stage**
    - You are near your target
    - ⇒ You must take chances by doing final dodge-tag
    - ⇒ You might want to distinguish between "corner", "side wall" and "open space" situations here

Assignment 3

Cheat sheet – Strategy – Catcher / Chaser

- Be sure not to pursue single target for a long time …
  - If you are unable to get from `early->mid` stage for a long time
  - If you are unable to get from `mid->final` stage for a long time
  - If your target manages to escape you and you switch from `final->mid` stage again

Assignment 3

Cheat sheet – Strategy – Catcher / Chaser

- Be sure to be aware who got tagged … and not only by you!
  - If someone got tagged, there is a good chance you can tag him as well
  - You can even try to count how much time it was needed to tag someone to be aware of the "skill" of your opponents

Assignment 3

Cheat sheet – Movement – Catcher / Chaser

`move.strafeTo(chasingLocation, escapeePlayer)`

- You should fix your focus to your prey while chasing
- Can be also used to "look around" during the chase, but that requires timing and won't probably work well

`move.dodge(chasingDirectionVector, false/true)`

- If your prey is near, you can try to quick dodge to it
- This will even work well during early stage of chase to quickly shorten the distance between you and your prey
- Be careful though as you might actually worsen your situation during final-stage of tagging as you can "miss dodge" your target
- `False/True` switches between `Single/Double` dodge modes

Assignment 3

Cheat sheet – Strategy – Runner / Escapee / Prey

- Your bot should try not to get cornered
- Your escape strategies typically distinguishes between 3 kinds of situations
  - Corner
    - You are in the corner
    - ⇒ Try quick successive double dodges or double jumps
    - ⇒ Then try to run for open-space position
  - Side-wall
    - Depending on the position of your chaser you should again
    - ⇒ Try quick successive double dodges or double jumps
    - ⇒ Then try to run for open-space position Open space
  - Open-space
    - You have a lot of space around you
    - ⇒ You should try to run in circles, but keep an eye on your chaser … you always have to decide which kind of circle-run you want to perform (clockwise / counterclockwise) preferably switching between those two as required by the situation

Assignment 3

Cheat sheet - Movement – Runner / Escapee / Prey

`move.strafeTo(escapeLocation, chaserPlayer)`

- Always use strafing and focus on the chaser to be sure to have up-to-date info about its position.
- Suitable for circle-runs

`move.dodge(escapeDirectionVector, true)`

- If in peril, try to perform double-dodge

`move.doubleJump()`

- … or double jump

Assignment 3

Extra Tournament Files

- Check the folder `TagBot/tournament`
- There are batch files to execute tournament matches
  - `match-best-2014.bat`
    - Performs match between the first 4 bots of the Tag! 2014
  - `match-123.bat`
    - Performs match between your bot and 1st, 2nd and 3rd bot of Tag! 2014
  - `match-456.bat`
    - Performs match between your bot and 4th, 5th and 6th bot of Tag! 2014

Assignment 3

Extra Tournament Files

- WARNING! You have to edit batch files first, to supply correct `UT2004_HOME` directory
- Alter the line

  ```
  set UT2004_HOME=d:\Games\UT2004-Devel
  ```

- To match your environment, e.g.

  ```
  set UT2004_HOME=c:\UT2004
  ```

Assignment 3

Extra Tournament Files

- WARNING! Execution of the batch file might override you bot/server ports within `UT2004_HOME\System\GameBots2004.ini`
  - You might bump into “connection refused” exceptions when trying to run your bot from `TagGame` of the template project

- Just restore original values within the `GameBots2004.ini` file, and restart a dedicated server:

```
[GameBots2004.BotDeathMatch]
BotServerPort=3000
ControlServerPort=3001
ObservingServerPort=3002
```

Assignment 3

Extra Tournament Videos

- Check the folder `TagBot/tournament-videos`
- There are several videos that might inspire you for coding Seeker/Runner behaviors

Send us finished assignment

Via e-mail:

- *Subject*
  - "Pogamut homework 2014 – 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 (Tuesday 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 (Tuesday practice lessons)
    - jakub.gemrot@gmail.com
  - Michal Bída (Monday practice lessons)
    - michal.bida@gmail.com