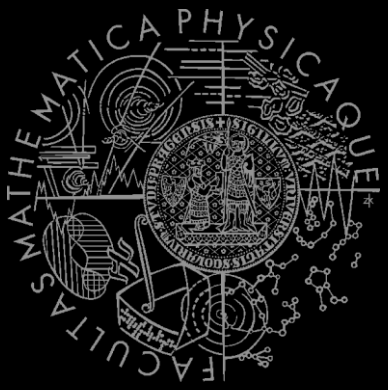
Faculty of Mathematics and Physics
Charles University
22nd February 2023



UT2004 bots made easy!

Pogamut 3

Lab 02 – ~~Running~~ Hunting Around

Wolves and Sheep Tournament!

Warm Up!

- **Fill the short test for this lab**

  7 minutes limit

  https://tinyurl.com/s53xjme2

  `0 vs. O, i vs. l vs. 1`

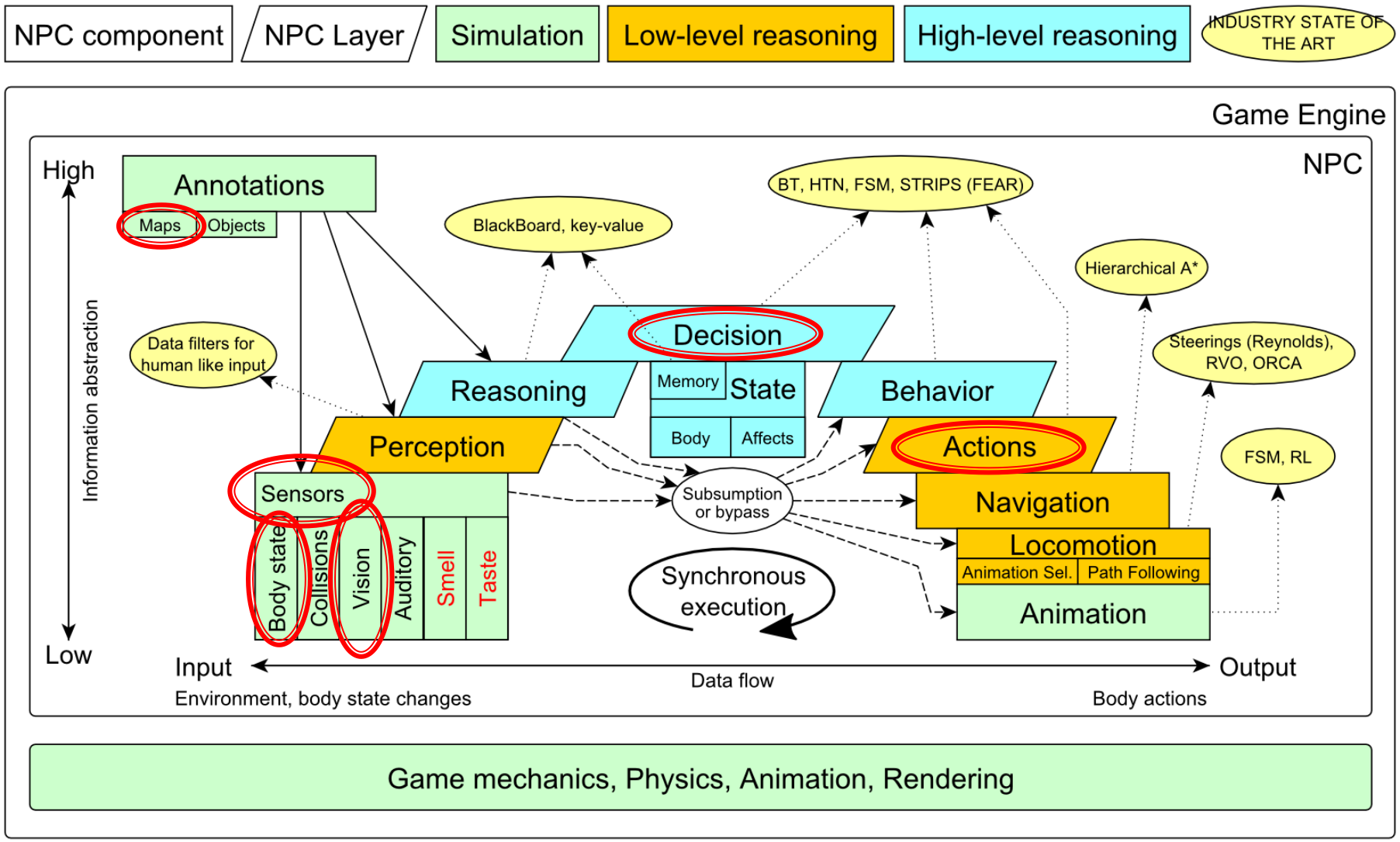
  Permanent link

  https://docs.google.com/forms/d/e/1FAIpQLSe9swXxlmFIuS5XhSIP9QexEyovLOyKnUcEUxMVP48LHfTpaw/viewform

Today's menu

1. **Big Picture**
2. Persistent Objects
3. Me and Other Players
4. Movement
5. Wolves and Sheep Homework
6. Wolves and Sheep Tournament

Big Picture
Today



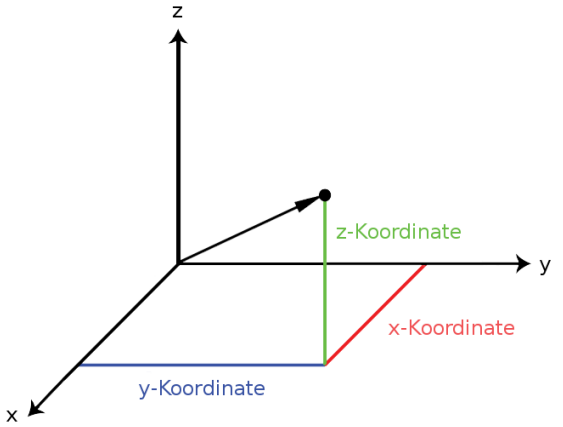
Basic sensors

Persistent Objects

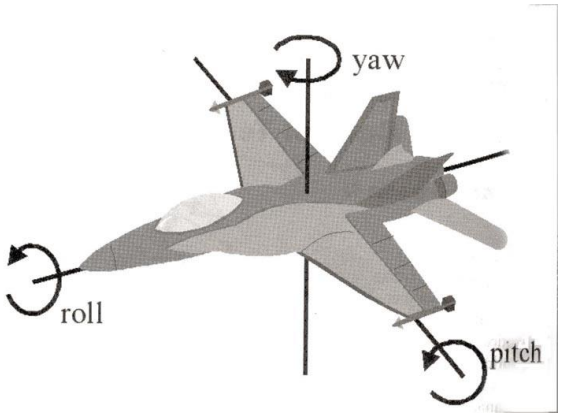
`Location, Rotation, Velocity, visibility`

Persistent Objects
Basic (descriptor) classes

- `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)

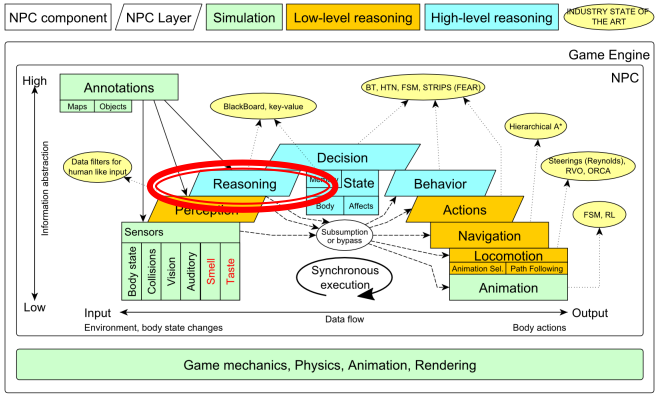


- `Velocity`
  - X, Y, Z vector
  - Length is speed in UT units (1UT ~ 1cm)
- All these objects are immutable

Persistent Objects

Basic attributes / attribute interfaces

- `Location -> ILocated`
  - Provides `getLocation()` method
- `Rotation -> IRotated`
  - Provides `getRotation()` method
- `Velocity -> ILocomotive`
  - Provides `getVelocity()` method
- `visibility -> IViewable`
  - Provides `isVisible()` method
  - Subjective information
- Every object is implementing at least `ILocated` and `IViewable` interfaces



Subjective information!

Persistent Objects
Unique identifiers

- Technically all persistent objects in Pogamut are implementing `IWorldObject` interface thus having following two methods:
  - `UnrealId getId()` => unique id of the object
  - `long getSimTime()` => timestamp of the last update
- Each unique id has single `UnrealId` instance
- Each unique object has single instance
  - There are no sneaky `clone()`s inside Pogamut library
  - Even deserialization heed to this rule
- There are three implications stemming from that
  - ⇒ You can compare ids and objects using plain ==, != opeators
  - ⇒ Ids and objects can be used in `Set<UnrealId>`, `Set<Player>`
  - ⇒ As well as keys in `Map<UnrealId, ?>`, `Map<Player, ?>`

IWorldView
Storage of IWorldObjects

- Low-level way to get a map of object of a concrete class
- `IWorldView ~ this.world`
  - `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)`

  …

Agent modules

Organizing information

`this.info`, `this.players`, …

Agent as set of modules

Sensors

- Agent Knowledge == set of „modules“
  - Each module specilized in doing something or providing information about something
- In Pogmaut, modules are organized around specific `InfoMessage`s, providing percept/reasoning routines associated with them to speed up coding of decision-making routines (behaviors / actions)
- Example: `players.getNearestVisibleEnemy()`

Agent as set of modules

Sensors

- Agent modules ~ low-level API façades
  - `AgentInfo ~ this.info ~ Self`
  - `Players ~ this.players ~ Player(s)`
- Advantages of encapsulation of API into methods instead objects:
  1. List of JavaDoced methods
  2. Functionality revolving around some class of entities at single place

=> Easier to read & understand the code

Basic sensors

Me and Other Players

`this.info, this.players`

Agent as set of modules

Sensors

- Class `AgentInfo ~ this.info`
  - Contains many information about your bot current state (except the inventory)
  - Technically reading information from `Self` message
  - Updated roughly **every 60ms**
- Class `Players ~ this.players`
  - Contains information about other players
  - Technically this is processing information from `Player` messages
  - By default updated **every 200ms**

Agent modules

Organizing actions

`this.move`, `this.shoot`, …

IAct
Low-level API and "movement" example

- All Pogamut action modules are using underlying `act` through which they send `CommandMessages`
- 4 classes affecting bot position/rotation
  - `Move`, `Jump`, `Dodge`, `TurnTo`
  - `this.act.act(new Move()…)`
  - `this.act.act(new Jump()…)`
  - `this.act.act(new Dodge()…)`
  - `this.act.act(new TurnTo()…)`

Movement Commands
Details

- `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
  - Can be performed by human players as well, just press one movement key twice (e.g.: press A A to dodge left)
- `TurnTo`
  - Used to rotate the bot while standing on the same location (not moving); you cannot combine this with `Move`!

Basic actions

Low-level Movement

```
this.move
```

Agent as set of modules
Actions

- Class `AdvancedLocomotion ~ this.move`
  - Wraps low-level `CommandMessage`s into methods
  - `move.moveTo(), move.strafeTo(), move.jump(), move.doubleJump(), move.turnXXX()`
  - Some simple vector math wrapped-in as well
    - `move.dodgeLeft(), move.dodgeRight(), …`

Homework 02

Wolves and Sheep

Gotta catch 'em all!

Homework 02

Gotta catch 'em all!

- You have to implement a `WolfBot`, that will be run `2x` and you will have to chase 12 `SheepBot`s down
  - BASE: catch at least 10 sheep in 120 seconds
  - ADVANCED: catch all 12 sheep in 60 seconds max, 5 adv. points
- Wolves-and-Sheep bot template
  - https://drive.google.com/file/d/18Y7WJeKGF4_Alx9uwjGn5RDdIjfqiHXi/view?usp=sharing
  - There you will find those two bots: `WolfBot` and `SheepBot`
- If you want advanced points, then deadline is next week!
  - If you do not meet homework deadline, you cannot gain advanced points!

Homework 02
Gotta catch 'em all!

- How to run?
  1. First you need to patch your UT2004 installation
     - Copy stuff from bot template `ut2004` directory onto the directory of your UT2004 installation
  2. Then start DM-TagMap using provided `startGamebotsDMServer-DM-TagMap.bat`
  3. Then start `SheepBot`
     - It will start 12 instances!
  4. Then start your `WolfBot`
     - It will start 2 instances!

Homework 02
CheatSheat

- See `TagMap` class for methods that provide you with environment awareness (distances from walls in this case)
- You will the most probably have to use `move.dodge(...)` !
- Do not forget you can move to some location while looking at some other using `move.strafe(...)`
- For the last few sheep, instead of trying to get lucky, you might actually want to synchronize your wolves
  - Use `instance` variable to distinguish between the alpha and the beta
- For fast collection filtering, see `DistanceUtils` static methods

- *Disclaimer: this list is not exhaustive…*

Homework 02

Submissions

Submissions will happen through Gdrive again.

Once you finish your homework, ZIP UP your project folder COMPLETELY (except the `target` folder) and upload the ZIP file to shared shared GDrive folder into `02-Wolves` directory.

Tournament 01

Wolves and Sheep

Gotta catch 'em all!

W&S Tournament

Chance to score advanced points!

- Catch all 12 sheep as fast as you can in 2 minutes!
- Each entry evaluated 20 times; the most sheep then faster time (sums) wins
- Tournament will be held in a week, see webpage!
- No cheating allowed … no shooting allowed, no bot speed reconfigurations allowed, etc.
- $1^{st}/2^{nd}/3^{rd}/4^{th}$ place receives 8/6/4/2 adv. points
- Good hunting!

Homework 02

Wolves and Sheep

Showcase

Questions?

I sense a soul in search of answers…

ASK AT DISCORD!

https://discord.gg/c49DHBJ