Faculty of mathematics and physics
Charles University in Prague
24st March 2015

UT2004 bots made easy!

# Pogamut 3

## Lecture 5 – Navigation

# Warm Up!

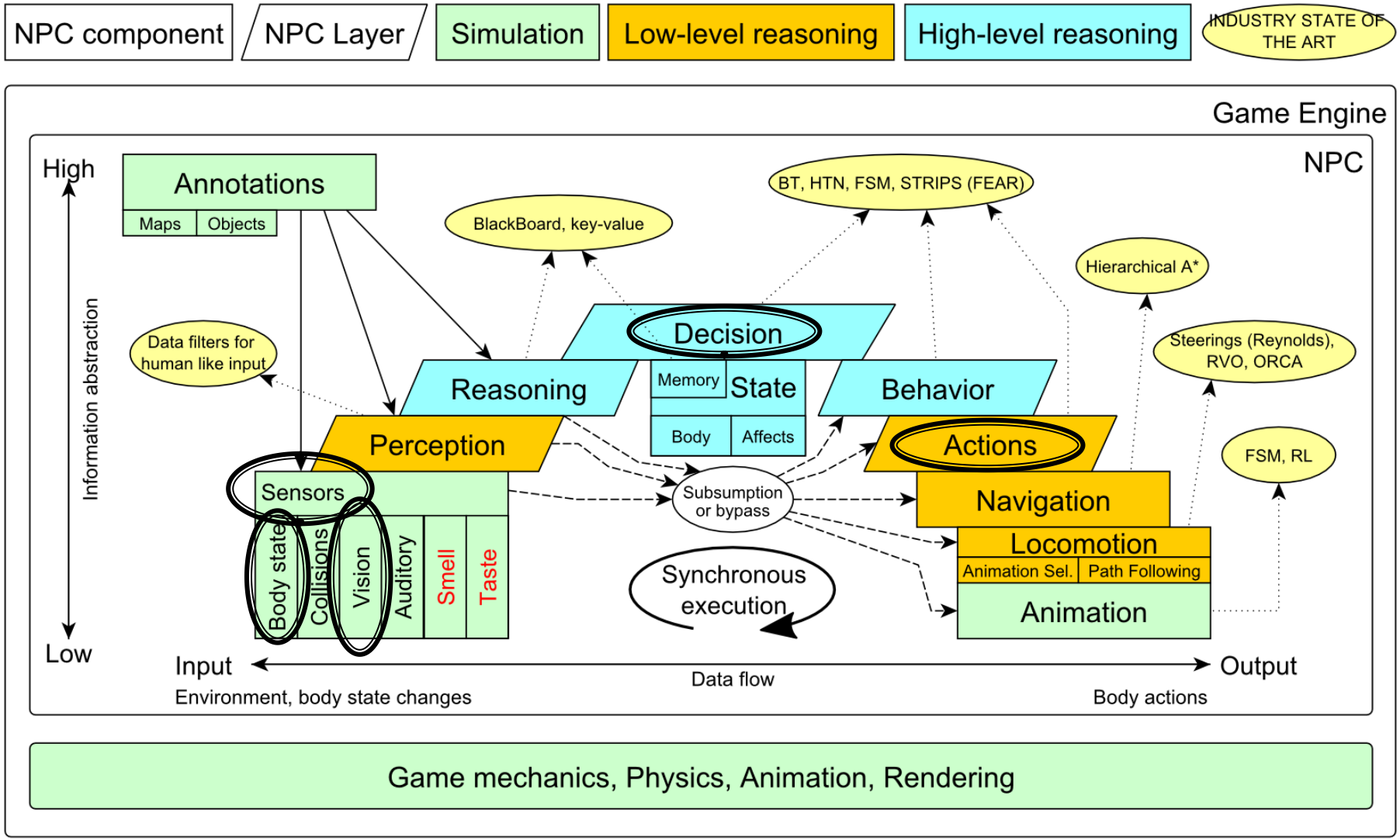

- Fill the short test for this lessons
  - 5 minutes limit
  - **XXX**
  - XXX

# Today's menu

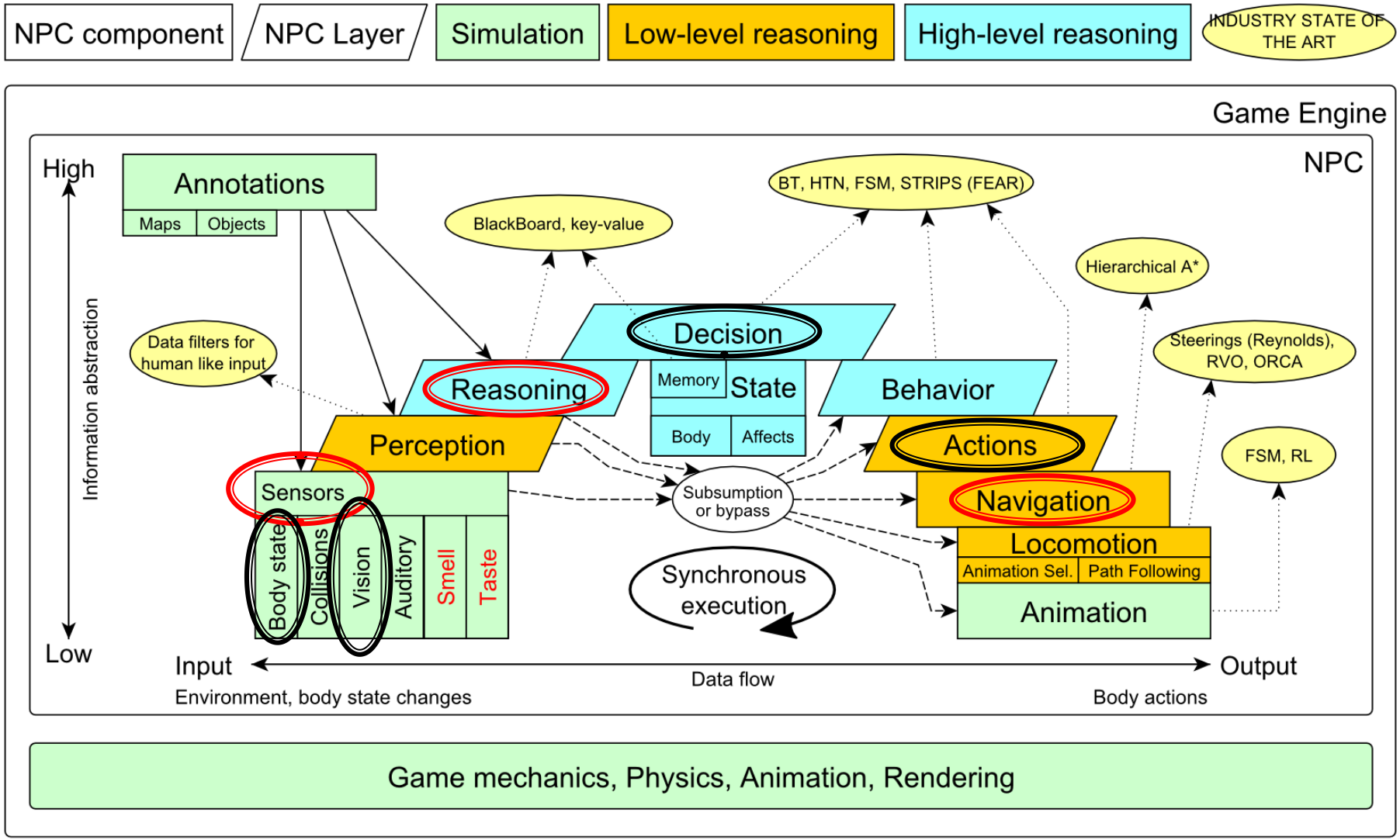

## Navigating inside UT2004

1. **Big Picture**
2. World Abstraction
3. Navigation

# Big Picture
## Already covered



# Big Picture
## Today



# Today's menu


## Navigating inside UT2004

1. Big Picture
2. **World Abstraction**
3. Navigation

# Pogamut World Abstraction

## Basics


***Objects (IWorldObject):***

- Player
- Item
- NavPoint
- Self
- IncomingProjectile

***Events (IWorldEvent):***

- HearNoise & HearPickup
- BotDamaged & BotKilled
- PlayerDamaged & PlayerKilled,
- Bumped
- GlobalChat

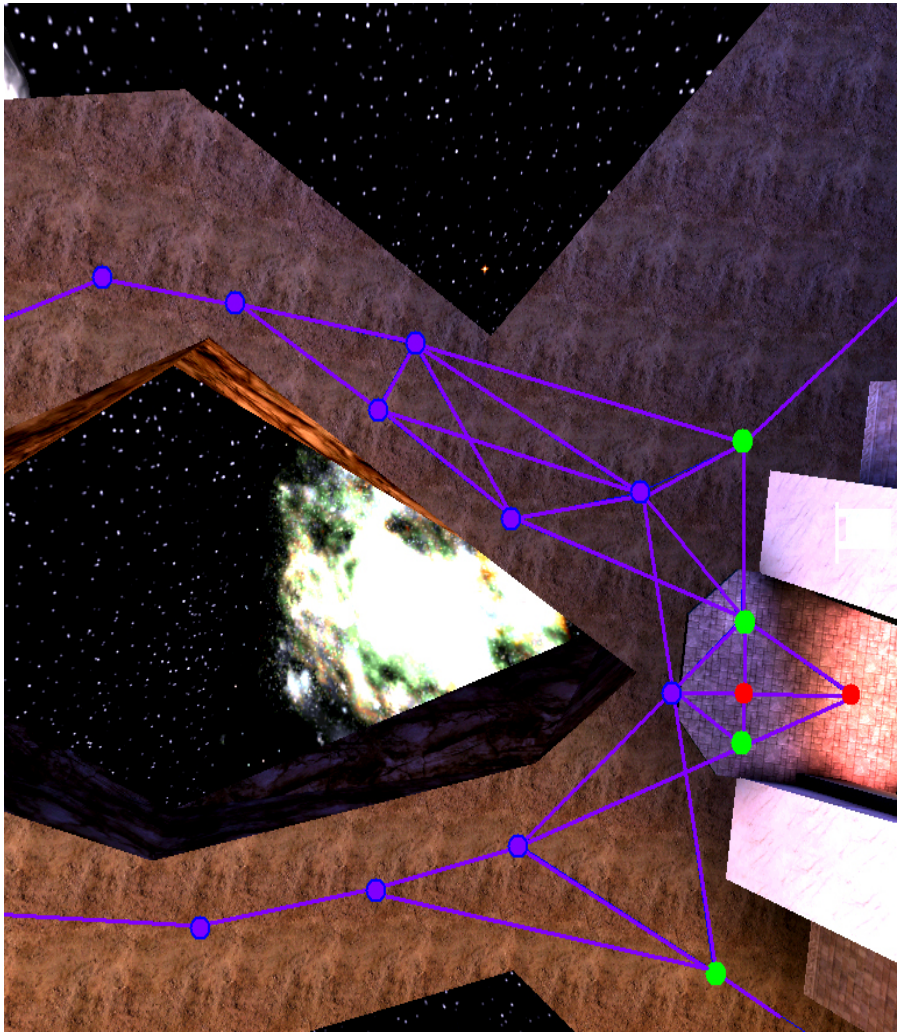
- Use modules, listeners and Pogamut helper classes!
  - this.players, this.items, this.info …
  - MyCollections, DistanceUtils

```
if (this.players.canSeePlayers()) {  …  }

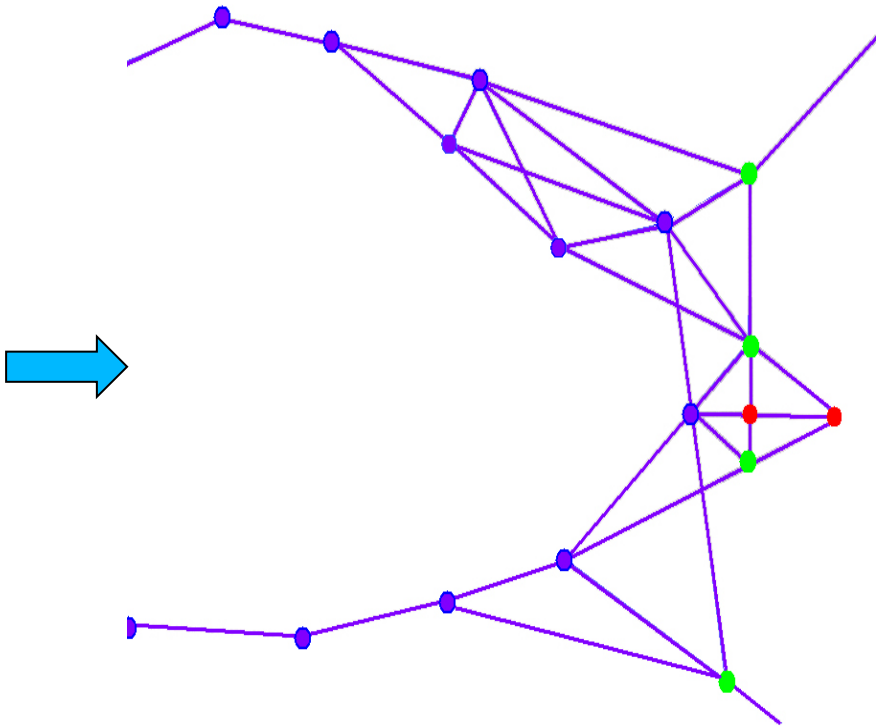
@EventListener(eventClass = GlobalChat.class)
public void chat(GlobalChat chatEvent) {
    …
}
```

# UT2004 World Abstraction

## Navigation graph



**#Navpoints in the map = 100 – 5000**



# UT2004 World Abstraction

## Underlaying classes – low level API


### *Classes of interest:*

NavPoint, NavPointNeighbourLink, **Item**

ILocated, Location, DistanceUtils

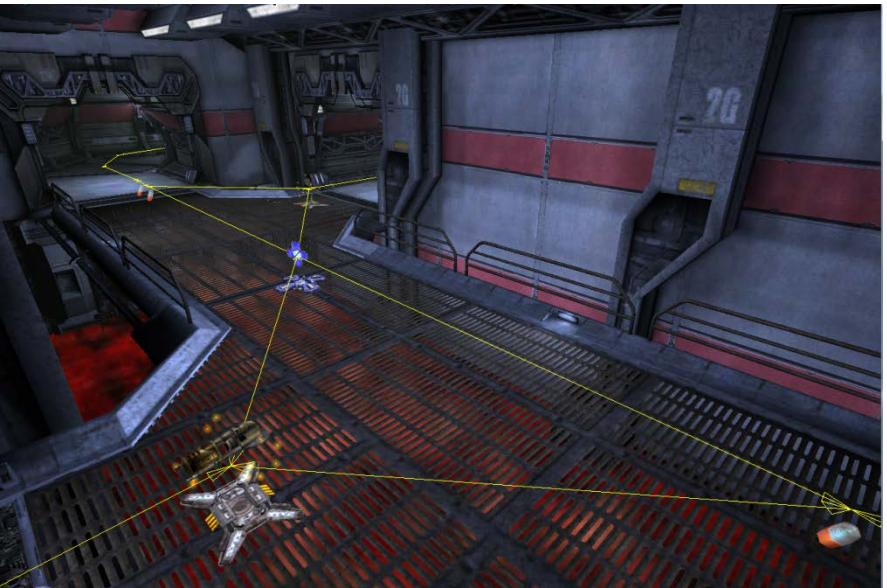
ItemType, ItemType.Category

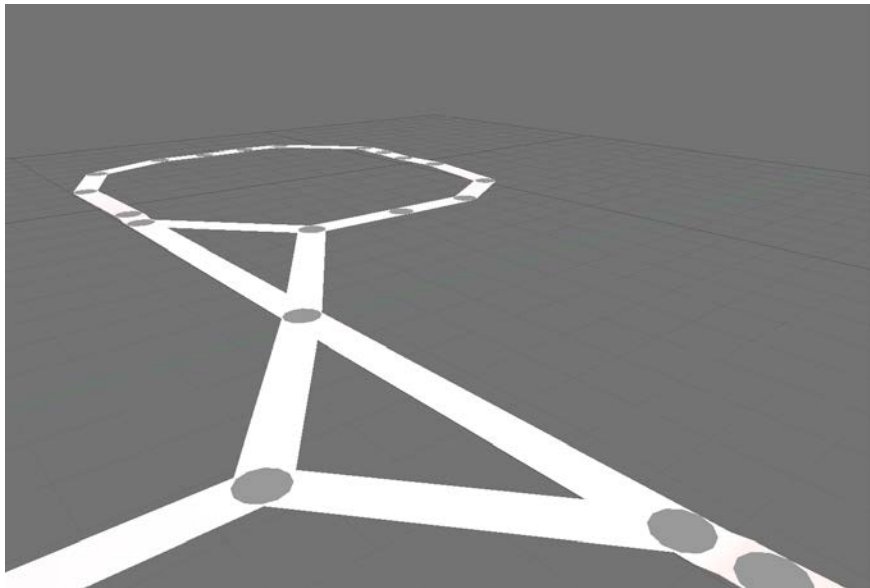
ItemDescriptor

### *Methods of interest:*

```
this.items.getAllItems(ItemType)
this.descriptors.getDescriptor(ItemType)
this.world.getAll(NavPoint.class)
this.world.getAll(Item.class)
NavPoint.getOutgoingEdges()
NavPoint.getIncomingEdges()
```

!!!





# UT2004 World Abstraction

## NavPoint/NeighbourLink types


- **NavPoint types**
  - JumpPad
  - Lift
  - Teleport
  - Door
  - PlayerStart
  - SnipingSpot
  - InventorySpot
  - …
- **Link flags**
  - Walk
  - Jump
  - Lift
  - Door
  - DoubleJump
  - …

# Today's menu


## Navigating inside UT2004

1. Big Picture
2. World Abstraction
3. **Navigation**

# Navigation

## Step by step


Navigation steps:

1. Decide where to go
2. Plan the path (list of navpoints)
3. Follow the path
   - Handle jumps&lifts along the way!
     - Do you know right constants?
   - World is non-deterministic, be sure to check how the action is executing!
     => `IStuckDetector` implementations

Don’t worry it’s already wrapped up ☺

# Navigation

## Step by step


1. Decide where to go (Decision making!)
   - `items.getSpawnedItems(ItemType)`
   - `navPoints.getNavPoints()`
   - `DistanceUtils.getNearest(…)`
   - `MyCollections.getRandom(…)`
   - `fwMap.getNearest(…)`
2. + 3. Plan and follow the path
   - UT2004Navigation (this.navigation)

# Navigation
## FloydWarshallMap


- Pogamut path planner uses **Floyd Warshall** algorithm ($O(n^3)$ !)
  - Used by `UT2004Navigation`
  - Access by `this.fwMap`
  - FW matrix is auto-initialized
- Methods of interest
  - `fwMap.getNearest…(…)`
    - Works the same as in `DistanceUtils`, except the distance is measured by the path length
    - Its ok to "spam" it (e.g. checking all items in each step), the nowadays computers can handle it

# Navigation

## UT2004Navigation


- Complete navigation wrapper
  - **UT2004Navigation**(…, UT2004PathExecutor, FloydWarshallMap, …) (this.navigation)
  - Handles both path planning & path following
  - Can be called repeatedly
  - Contains this.pathExecutor, this.fwMap
- Main methods
  - navigation.navigate(…)
  - navigation.isNavigating()
  - navigation.stopNavigation()
- Uses
  - FloydWarshallMap (this.fwMap)
  - StuckDetectors
  - UT2004PathExecutor

# Navigation

## Modifying the navigation graph


- NavigationGraphBuilder
  - Access by `this.navBuilder`
- Methods of interest
  - `navBuilder.removeEdge(…)`
  - `navBuilder.removeEdgesBetween(…)`
- If you use `navBuilder` in **botInitalized** method, everything will be applied automatically
  - Otherwise, call `fwMap.refreshPathMatrix()`
    - $O(n^3)$ !!

# Navigation
## StuckDetectors


- Navigation Uses 3 stuck detectors
- **AccUT2004TimeStuckDetector(bot, 3000)**
  - if the bot does not move for 3 seconds consider it is stuck (check small velocity delta)
- **AccUT2004PositionStuckDetector()**
  - watch over the position history of the bot, if the bot does not move sufficiently enough, consider that it is stuck
  - DEFAULT_HISTORY_LENGTH, DEFAULT_MIN_DIAMETER, DEFAULT_MIN_Z
- **AccUT2004DistanceStuckDetector()**
  - counts how many times the bot was getting closer to the target and how many times it was getting farther (if it oscillates more than two times -> STUCK)

# Navigation

## Listening for navigation events


- With a FlagListener! Add one with method addStrongNavigationListener

```
this.navigation.addStrongNavigationListener(
  new FlagListener<NavigationState>() {
     @Override
     public void flagChanged(NavigationState changedValue){
         switch (changedValue) {
             case STUCK:
                 break;
             case STOPPED:
                 break;
             case TARGET_REACHED:
                 break;
             case PATH_COMPUTATION_FAILED:
                 break;
             case NAVIGATING:
                 break;
         }
     }
  });
```

# Navigation
## Path following hell


- **UT2004PathExecutor**
- Custom Pogamut path following code
  - Heavily tweaked for UT2004 and game update frequency 4 Hz (250 ms per synchronous batch)
- The good
  - Works decently on non-complex maps
  - You don't have to do it yourself
- The bad
  - Has problems handling complex links
  - Spaghetti code

# Navigation

## Stuck detection details


- Inside `UT2004PathExecutorStuckState`
  - Who has detected the stuck
  - Which `NavPointNeighboutLink` bot failed to traverse
- Version: 3.5.1-SNAPSHOT and later

```
this.pathExecutor.getState().addStrongListener(
    new FlagListener<IPathExecutorState>() {
        @Override
        public void flagChanged(IPathExecutorState event) {
            switch (changedValue.getState()) {
                case STUCK:
                    UT2004PathExecutorStuckState
                    stuckDetails =
                            (UT2004PathExecutorStuckState)
                            event;
                    log.info("STUCK by: " +
                         stuckDetails.getStuckDetector().getClass()
                        .getSimpleName()
                    );
                    …
                    break;
                …
            }
    }
});
```

# NavMesh Navigation

## The most reliable navigation

# NavMesh Navigation

## The most reliable navigation


- Combination of NavMesh + NavGraph
  - Contains Off-Mesh Connections
    - Former NavGraph links that connects non-adjacent meshes, which are not completely "within" navmesh

  ⇒ All jump links are typically present within the "NavMeshGraph"

- **NavMeshNavigation**
  - Implements `IUT2004Navigation`
    - Same interface as `UT2004Navigation`
  - Usable only iff NavMesh static data for the concrete map is present!
    - Check it via `navMeshModule.isInitialized()`

# NavMesh Navigation

## The most reliable navigation


- NavMesh Static Data
  - Expected to be inside `./navmesh` folder (project root dir)
  - Downloadable from `svn://artemis.ms.mff.cuni.cz/pogamut/trunk/project/Addons/UT2004NavMeshTools/04-NavMeshes`
    - Text files in the form of <map-name>.navmesh
  - .navmesh file gets combined with NavGraph during bot startup and saved within .processed file
  - If you are going to play with `navBuilder`, you will have to tell `navMeshModule` that you want the NavMesh to be reloaded and recombined with your changed version of NavGraph

# NavMesh Navigation

## The most reliable navigation


- Correct way of NavMesh reloading

```
@Override
public void mapInfoObtained() {
    mapTweaks.register("DM-1on1-Albatross", new IMapTweak() {

        @Override
        public void tweak(NavigationGraphBuilder builder) {
            // alter the navgraph here
        }

    });
    navMeshModule.setReloadNavMesh(true);
}
```

# Assignment 5
## Navigation Bot


- **NavTesterBot**
  - Download template
  - Extend the bot so it endlessly runs between two given points
    - Use Respawn command
  - See the "navigation bug sheet" for DM-1on1-Roughinery-FPS
  - Find "bugs" assigned to you (via Task No.)
  - Minimalize each problem to the "shortest" path
  - If the bug is caused by misplaced point/link use navBuilder to fix it
  - Fix at least 1 hard bug within the navigation code
    - MUST BE CORRECTLY COMMENTED
      - `// OLD CODE:`
      - `// PROBLEM:`
      - `// NEW CODE:`

- *10 points*
  - *Extra 10 points if you manage to fix 3 hard bugs within navigation code*

# Assignment 5
## Cheatsheet


- Deciding where to go
  - `navPoints.getNavPoint()`
  - `DistanceUtils…`
- Navigation module
  - `this.navigation.navigate(…)`
  - `this.navigation.isNavigating()`
- Stuck listening
  - ```
    this.navigation
          .addStrongNavigationListener(
    new FlagListener<NavigationState>() { … })
    ```
- Info about the bot
  - `this.info.getLocation()`
  - `this.info.atLocation(ILocated)`
    ```

# Send us finished assignment


Via e-mail:

- *Subject*
  - "Pogamut homework 2015 – 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