



Game Oriented Multi Agent System, based on Jade

Práctica TAIA , FISS

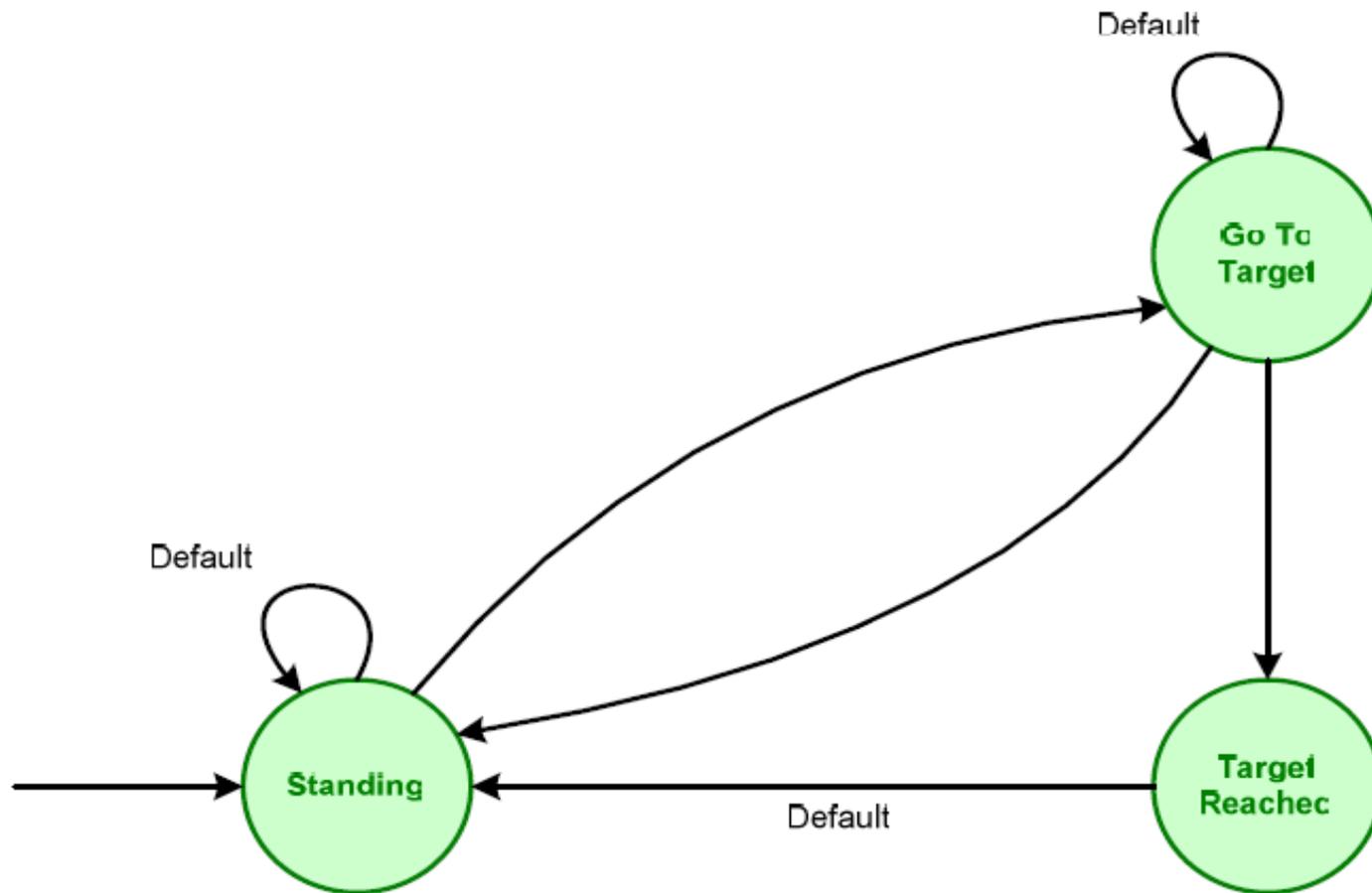
*Toni Barella, DSIC – UPV
tbarella@dsic.upv.es*

<http://www.dsic.upv.es/users/ia/sma/tools/jgomas/>

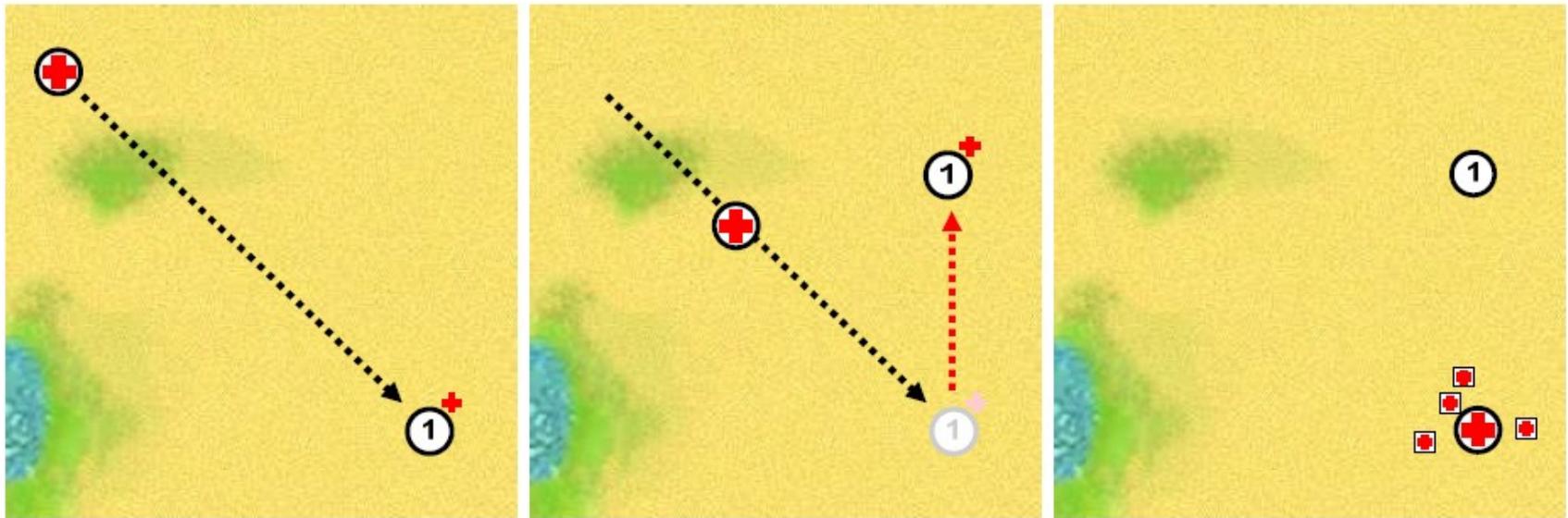
Índice

- **Bucle de Ejecución**
- Ejecución: Caso 2
- Ejecución: Caso 3
- Consideraciones
- Trabajo a realizar, Ideas y Reglas básicas.

Bucle de ejecución



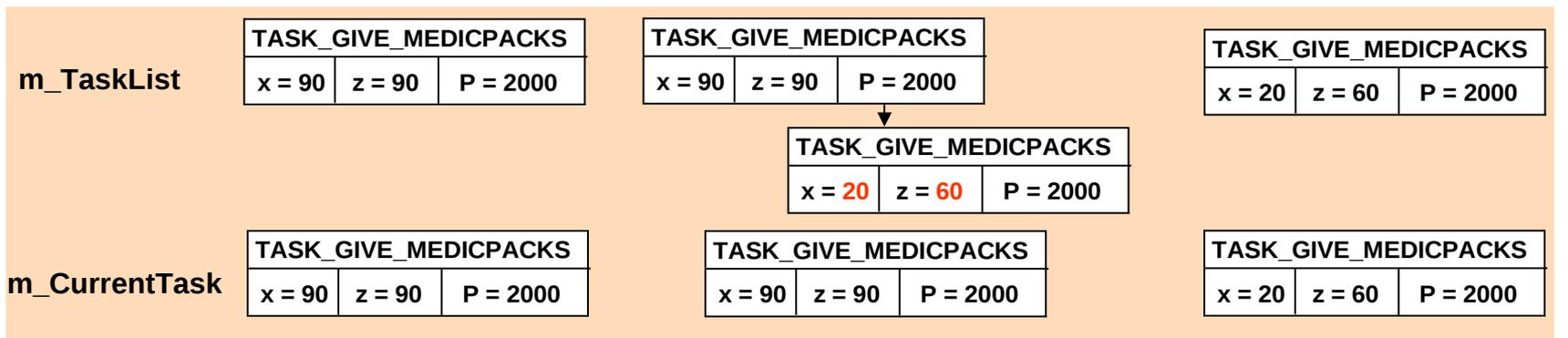
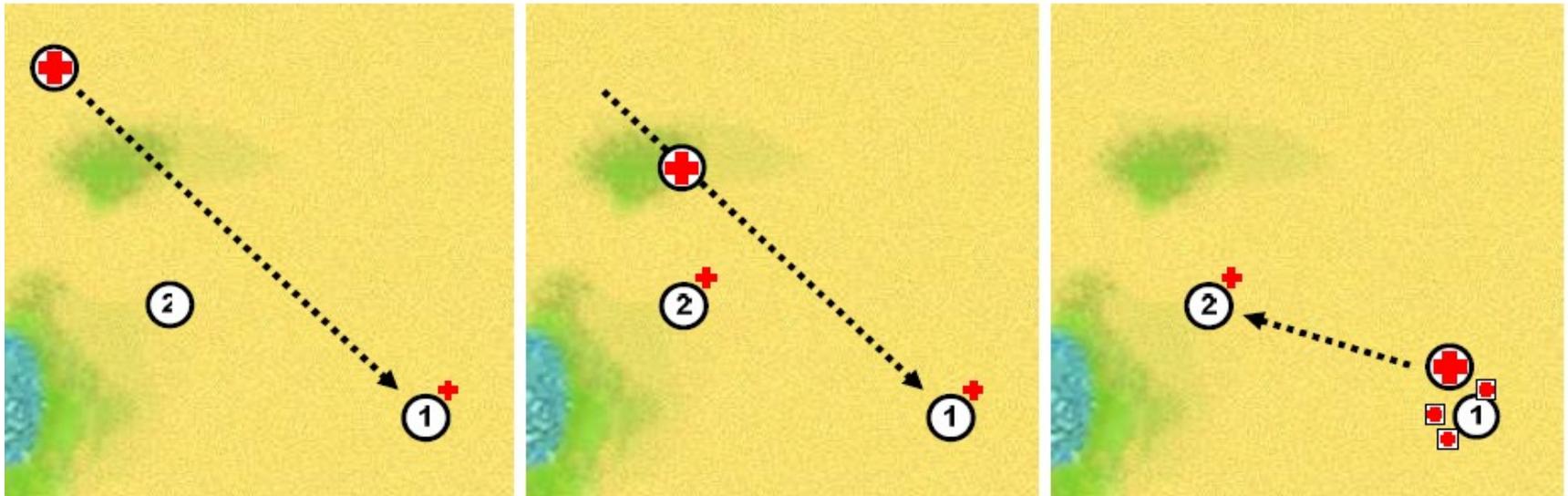
Ejecución: Caso 2



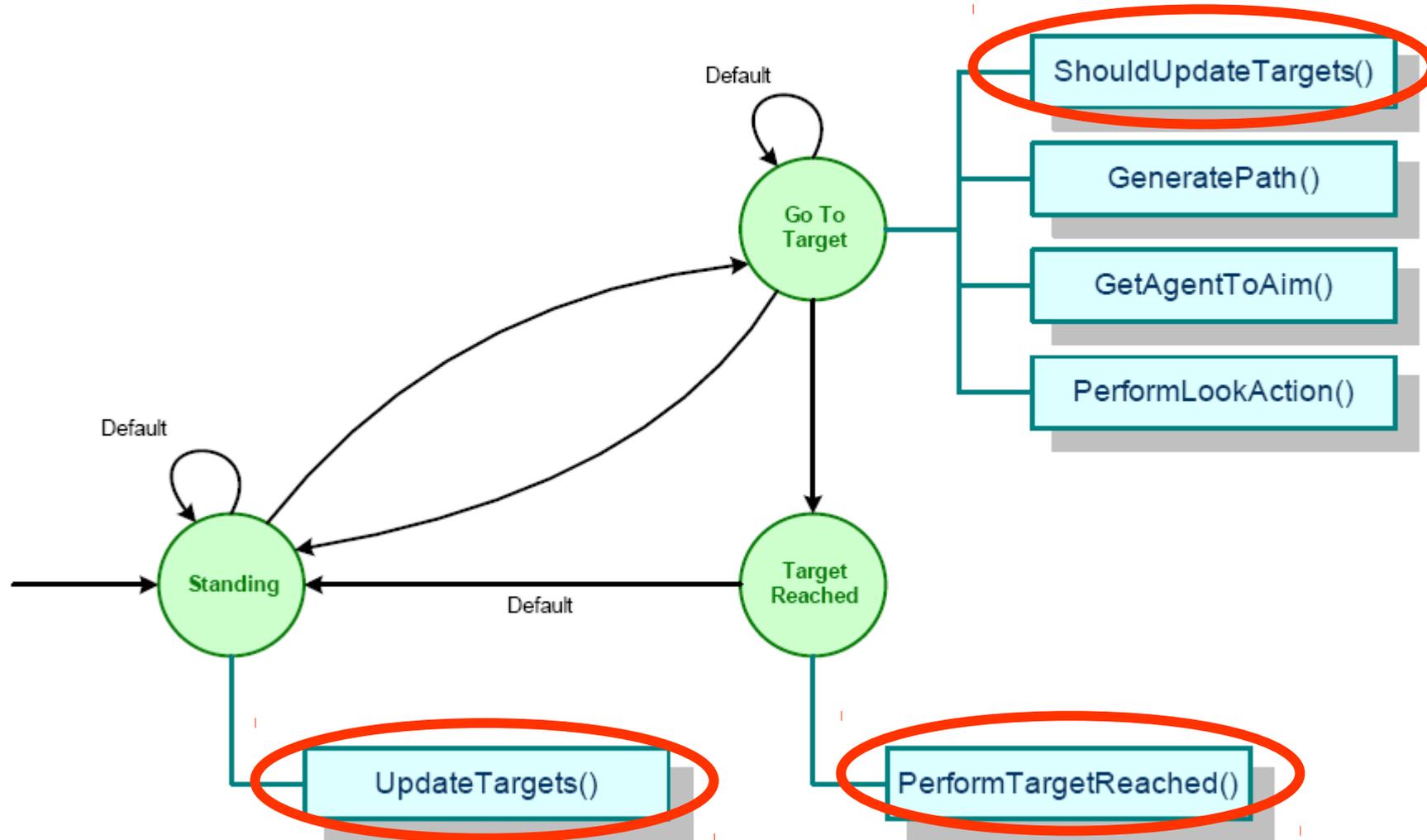
| | | | |
|---------------|----------------------|----------------------|----------|
| m_TaskList | TASK_GIVE_MEDICPACKS | TASK_GIVE_MEDICPACKS | |
| | x = 90 | z = 90 | P = 2000 |
| m_CurrentTask | TASK_GIVE_MEDICPACKS | TASK_GIVE_MEDICPACKS | -- |
| | x = 90 | z = 90 | P = 2000 |



Ejecución: Caso 3



Consideraciones



Consideraciones

Umbrales

- Cada agente dispone de una variable umbral: **m_Treshold**
- Permite definir límites de:
 - Salud
 - Munición
- Una vez excedidos se lanza:
 - CallForMedic()**
 - CallForAmmo()**



Consideraciones

Umbrales

- Además también se ejecuta el método :
PerformThresholdAction()
- Métodos de acceso a los umbrales:
 - Cota inferior de Salud
 - **void SetHealth (int _iHealth)**
 - **int GetHealth()**
 - Cota inferior de Munición
 - **void SetAmmo (int _iAmmo)**
 - **int GetAmmo()**

CBasicTroop (I): Métodos finales

- `final int GetHealth ()`
- `final int GetAmmo ()`
- `final int GetStamina ()`
- `final void UseStamina ()`
- `final int GetPower ()`
- `final void UsePower ()`
- `final void AddServiceType (String _sServiceType)`
- `final boolean CheckStaticPosition ()`
- `final boolean CheckStaticPosition (double _x, double _z)`
- `final void AddTask (int _tTypeOfTask, AID _Owner, String _sContent)`
- `final void AddTask (int _tTypeOfTask, AID _Owner, String _sContent, int _iPriority)`
- `final void Look ()`
- `final boolean Shot (int _iShotNum)`
- `final void PerformAimAction ()`
- `final boolean HaveAgentToShot ()`



CBasicTroop (II): M. Sobrecargables

- void CallForMedic ()
- void CallForAmmo ()
- void CallForBackup ()
- void UpdateTargets ()
- boolean **ShouldUpdateTargets** ()
- void ObjectivePackTaken ()
- void SetUpPriorities ()
- void PerformNoAmmoAction ()
- void PerformTargetReached (CTask _CurrentTask)
- void GenerateEscapePosition ()
- boolean GeneratePath ()
- void CreateControlPoints ()
- void PerformThresholdAction ()
- void PerformInjuryAction ()
- boolean GetAgentToAim ()
- void PerformLookAction ()



CBasicTroop (III): Atributos (1)

- `int` `m_eTeam`
- `int` `m_eClass`

- `AID` `m_Manager`
- `Hashtable` `m_TaskList`
- `CTask` `m_CurrentTask`
- `int` `m_TaskPriority` []

- `ArrayList` `m_FOVObjects`
- `CSight` `m_AimedAgent`
- `boolean` `m_bObjectiveCarried`

- `Vector3D` `m_ControlPoints` []
- `int` `m_iControlPointsIndex`
- `Vector3D` `m_AStarPath` []
- `int` `m_iAStarPathIndex`



CBasicTroop (III): Atributos (2)

- CThreshold m_Threshold
- CMobile m_Movement
- CTerrainMap m_Map

- String m_sMedicService
- String m_sAmmoService
- String m_sBackupService

- int m_iSoldiersCount,
- int m_iMedicsCount
- int m_iFieldOpsCount
- int m_iTeamCount



Métodos específicos

■ Métodos finales

| | |
|------------------|---|
| Cmedic | <code>final int CreateMedicPack ()</code> |
| CFieldOps | <code>final int CreateAmmoPack ()</code> |
| Csoldier | No hay |

■ Métodos sobrecargables

| | |
|------------------|---|
| Cmedic | <code>void SetUpPriorities ()</code> <code>boolean checkMedicAction (String _sContent)</code> |
| CFieldOps | <code>void SetUpPriorities ()</code> <code>boolean checkAmmoAction (String _sContent)</code> |
| Csoldier | <code>void SetUpPriorities ()</code> <code>boolean checkBackupAction (String _sContent)</code> |



Trabajo a Realizar

- Diseñar e implementar un equipo aliados y otro del eje, de **10 agentes** cada uno.
- La distribución de agentes (médicos, soldados y fieldops) que mejor consideréis.
- Jugar a **capturar la bandera**, en map_01 (sin obstáculos).
- **Objetivo** ganar en cualquier situación a los equipos suministrados.



Trabajo a Realizar

- Documentación a entregar:
 - Pequeña memoria, indicando:
 - las principales ideas de mejora aplicadas al equipo, así como unas breves conclusiones sobre los resultados obtenidos.
 - Composición de los destacamentos.
 - El código, comentado y documentado, indicando qué partes se han añadido y/o modificado.
 - Este código debe seguir unas mínimas normas de estilo: tabulado, comentado, y usando notación húngara.



■ Notación Húngara: Identificadores

- Ejemplo: `CString m_sNameFirst;`

- Prefijo Calificador

- Prefijo:
 - Siempre en minúsculas
 - Indica el tipo del dato

- Calificador:
 - Resto del nombre, indica el uso de la var.
 - Empieza por mayúscula, para delimitarlo del prefijo
 - Si contienen más de una palabra, la inicial de cada una va en mayúscula. Ej.: `EstoEsUnIdMuyLargo;`

■ Prefijos comunes

- **s** ó **str**: *String*
- **c**: Carácter
- **i** ó **n (ó e)**: Entero
- **f**: *Float*
- **d**: Doble
- **b**: *Boolean*
- **l**: *Long*
- **C**: Clase
- **m_**: Identificador de un miembro de una clase
- **v**: *Void*, sin tipo



Ideas

- *Coordinación del equipo atacante sin comunicación:*
 - Cada agente ha de seguir a un agente de su propio equipo que se encuentra “más alejado de”/”más cerca de” o es más importante que él.
- *Coordinación del equipo defensor con comunicación:*
 - Cada agente ha de patrullar pasando por los puntos de patrulla del resto de los agentes
- Cuando un agente necesite un servicio que él mismo suministra, no debe solicitarlo fuera, sino que se lo pueda suministrar él mismo.



Reglas Básicas

- No se puede consultar/solicitar información del sistema sobre el bando contrario que no sea suministrada por el entorno.
- No puede existir comunicación entre agentes que no sea usando FIPA ACL y de acuerdo a la especificación proporcionada.