Everything is fine

Simulation de protocoles d'évacuation & gestion de panique

Édouard BREUILLÉ
Célia ROUQUAIROL

Master 2 IMAGINA - Université des sciences

Le niveau micro

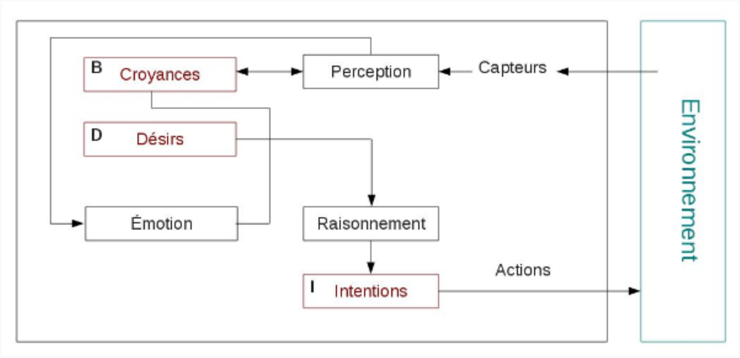
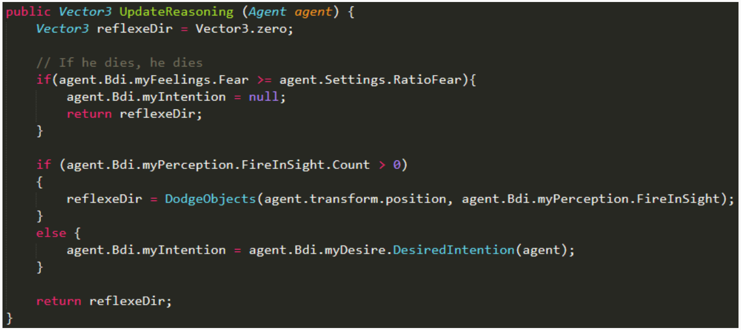


Figure 1: Le modèle BDI

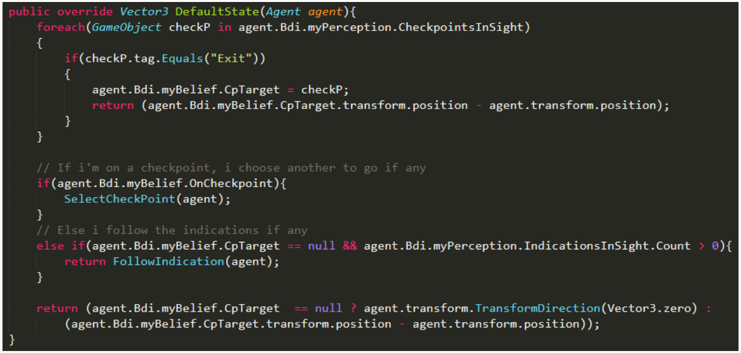
Le niveau micro



```
public Vector3 UpdateReasoning (Agent agent) {
    Vector3 reflexeDir = Vector3.zero;

    // If he dies, he dies
    if(agent.Bdi.myFeelings.Fear >= agent.Settings.RatioFear){
        agent.Bdi.myIntention = null;
        return reflexeDir;
    }

    if (agent.Bdi.myPerception.FireInSight.Count > 0)
    {
        reflexeDir = DodgeObjects(agent.transform.position, agent.Bdi.myPerception.FireInSight);
    }
    else {
        agent.Bdi.myIntention = agent.Bdi.myDesire.DesiredIntention(agent);
    }

    return reflexeDir;
}
```

Figure 2: Raisonnement

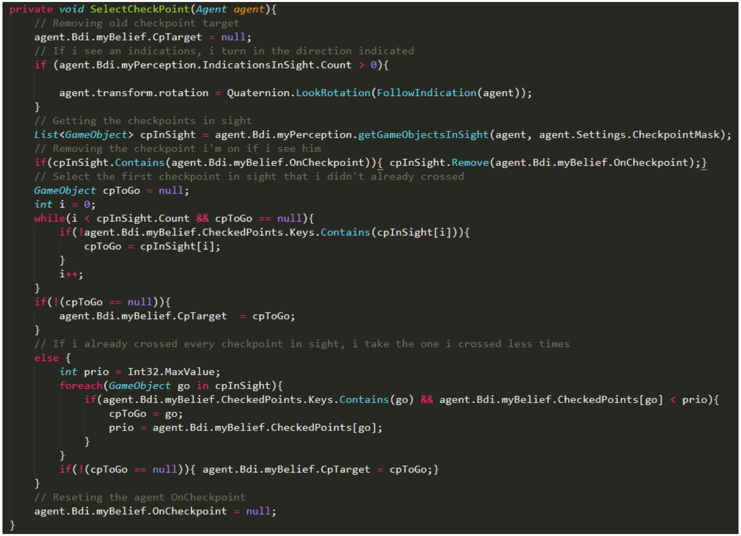
Le niveau micro



```
public override Vector3 DefaultState(Agent agent){
    foreach(GameObject checkP in agent.Bdi.myPerception.CheckpointsInSight)
    {
        if(checkP.tag.Equals("Exit"))
        {
            agent.Bdi.myBelief.CpTarget = checkP;
            return (agent.Bdi.myBelief.CpTarget.transform.position - agent.transform.position);
        }
    }

    // If i'm on a checkpoint, i choose another to go if any
    if(agent.Bdi.myBelief.OnCheckpoint){
        SelectCheckPoint(agent);
    }
    // Else i follow the indications if any
    else if(agent.Bdi.myBelief.CpTarget == null && agent.Bdi.myPerception.IndicationsInSight.Count > 0){
        return FollowIndication(agent);
    }

    return (agent.Bdi.myBelief.CpTarget  == null ? agent.transform.TransformDirection(Vector3.zero) :
        (agent.Bdi.myBelief.CpTarget.transform.position - agent.transform.position));
}
```

Figure 3: L'intention de sortir

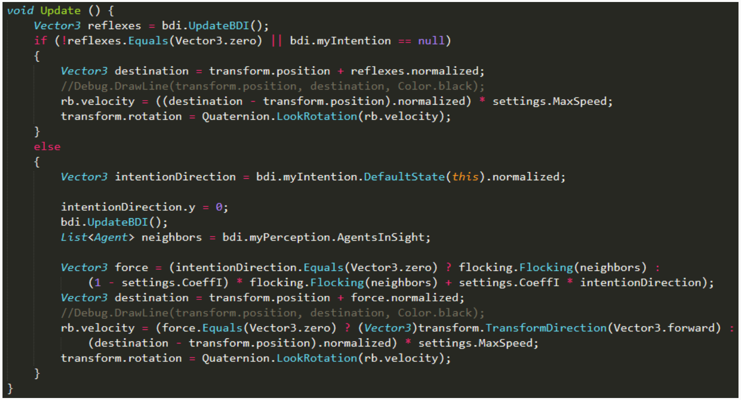
Le niveau micro



```
private void SelectCheckPoint(Agent agent){
    // Removing old checkpoint target
    agent.Bdi.myBelief.CpTarget = null;
    // If i see an indications, i turn in the direction indicated
    if (agent.Bdi.myPerception.IndicationsInSight.Count > 0){

        agent.transform.rotation = Quaternion.LookRotation(FollowIndication(agent));
    }
    // Getting the checkpoints in sight
    List<GameObject> cpInSight = agent.Bdi.myPerception.getGameObjectsInSight(agent, agent.Settings.CheckpointMask);
    // Removing the checkpoint i'm on if i see him
    if(cpInSight.Contains(agent.Bdi.myBelief.OnCheckpoint)){ cpInSight.Remove(agent.Bdi.myBelief.OnCheckpoint);}
    // Select the first checkpoint in sight that i didn't already crossed
    GameObject cpToGo = null;
    int i = 0;
    while(i < cpInSight.Count && cpToGo == null){
        if(!agent.Bdi.myBelief.CheckedPoints.Keys.Contains(cpInSight[i])){
            cpToGo = cpInSight[i];
        }
        i++;
    }
    if(!(cpToGo == null)){
        agent.Bdi.myBelief.CpTarget  = cpToGo;
    }
    // If i already crossed every checkpoint in sight, i take the one i crossed less times
    else {
        int prio = Int32.MaxValue;
        foreach(GameObject go in cpInSight){
            if(agent.Bdi.myBelief.CheckedPoints.Keys.Contains(go) && agent.Bdi.myBelief.CheckedPoints[go] < prio){
                cpToGo = go;
                prio = agent.Bdi.myBelief.CheckedPoints[go];
            }
        }
        if(!(cpToGo == null)){ agent.Bdi.myBelief.CpTarget = cpToGo;}
    }
    // Reseting the agent OnCheckpoint
    agent.Bdi.myBelief.OnCheckpoint = null;
}
```

Figure 4: Choix à un croisement

Le niveau micro



```
void Update () {
    Vector3 reflexes = bdi.UpdateBDI();
    if (!reflexes.Equals(Vector3.zero) || bdi.myIntention == null)
    {
        Vector3 destination = transform.position + reflexes.normalized;
        //Debug.DrawLine(transform.position, destination, Color.black);
        rb.velocity = ((destination - transform.position).normalized) * settings.MaxSpeed;
        transform.rotation = Quaternion.LookRotation(rb.velocity);
    }
    else
    {
        Vector3 intentionDirection = bdi.myIntention.DefaultState(this).normalized;

        intentionDirection.y = 0;
        bdi.UpdateBDI();
        List<Agent> neighbors = bdi.myPerception.AgentsInSight;

        Vector3 force = (intentionDirection.Equals(Vector3.zero) ? flocking.Flocking(neighbors) :
            (1 - settings.CoeffI) * flocking.Flocking(neighbors) + settings.CoeffI * intentionDirection);
        Vector3 destination = transform.position + force.normalized;
        //Debug.DrawLine(transform.position, destination, Color.black);
        rb.velocity = (force.Equals(Vector3.zero) ? (Vector3)transform.TransformDirection(Vector3.forward) :
            (destination - transform.position).normalized) * settings.MaxSpeed;
        transform.rotation = Quaternion.LookRotation(rb.velocity);
    }
}
```

Figure 5: Le comportement de l'agent