NPC centralized control – Behavior Trees / AI

"Tick and Think"

SERVER SIDE (partial):

```
public class NPCcontroller
{
  BehaviorTree bt = new BehaviorTree(BTCompositeType.SELECTOR);
  . . .
  public void start ()
  { thinkStartTime = System.nanoTime();
   tickStateTime = System.nanoTime();
   lastThinkUpdateTime = thinkStartTime;
   lastTickUpdateTime = tickStartTime;
   setupNPC();
   setupBehaviorTree();
   npcLoop();
  }
```
- -
```
  public void setupNPC()
  { npc = new NPC();
   npc.randomizeLocation(rn.nextInt(50),rn.nextInt(50));
  }
```
- -
```
  public void npcLoop()
  { while (true)
   { long currentTime = System.nanoTime();
    float elapsedThinkMilliSecs = (currentTime-lastThinkUpdateTime)/(1000000.0f);
    float elapsedTickMilliSecs = (currentTime-lastTickUpdateTime)/(1000000.0f);

    if (elapsedTickMilliSecs >= 50.0f)          // "TICK"
    {   lastTickUpdateTime = currentTime;
       npc.updateLocation();
       server.sendNPCinfo();
    }

    if (elapsedThinkMilliSecs >= 500.0f)        // "THINK"
    {   lastThinkUpdateTime = currentTime;
       bt.update(elapsedMilliSecs);
    }
    Thread.yield();
  }}
```
- -
```
  public void setupBehaviorTree()
  { bt.insertAtRoot(new BTSequence(10));
   bt.insertAtRoot(new BTSequence(20));
   bt.insert(10, new OneSecPassed(this,npc,false));
   bt.insert(10, new GetSmall(npc));
   bt.insert(20, new AvatarNear(server,this,npc,false));
   bt.insert(20, new GetBig(npc));
}}
```

```
public class AvatarNear extends BTCondition
{
 public AvatarNear(GameServerTCP s, NPCcontroller c, NPC n, boolean toNegate)
 { super(toNegate);
  server = s;
  npcc = c;
  npc = n;
 }

 protected boolean check()
 { server.sendCheckForAvatarNear();
  return npcc.getNearFlag();
}}
```

```
public class GetSmall extends BTAction
{
  public GetSmall(NPC n)  { npc = n; }

  protected BTStatus update(float elapsedTime)
  { npc.getSmall();
   return BTStatus.BH_SUCCESS;
}}
```

```
public class GetBig extends BTAction
{
  public GetBig(NPC n)  { npc = n; }

  protected BTStatus update(float elapsedTime)
  { npc.getBig();
   return BTStatus.BH_SUCCESS;
}}
```

```
public class OneSecPassed extends BTCondition
{
  public OneSecPassed(NPCcontroller c, NPC n, boolean toNegate)
  { super(toNegate);
   npcc = c;
   npc = n;
   lastUpdateTime = System.nanoTime();
  }

  protected boolean check()
  { float elapsedMilliSecs = (System.nanoTime()-lastUpdateTime)/(1000000.0f);
   if ((elapsedMilliSecs >= 1000.0f) && (npc.getSize()==2.0))
   { lastUpdateTime = System.nanoTime();
    npcc.setNearFlag(false);
    return true;
   }
   else return false;
}}
```