Project 1: Behavior Trees

# General Questions

##### Why can’t I compile with VS2017 in Build.bat?

As of April 2017. Visual Studio 2017 no longer adds its path to environment variables anymore. Since OS can’t locate “%VS150COMNTOOLS%..\IDE\devenv”, command line batch script won’t be able to run successfully. Do not know if it’s a bug or an intension by Microsoft. But in order to run Build.bat, you need to add the path manually. For example:

set VS150COMNTOOLS=C:\Program Files (x86)\Microsoft Visual Studio\2017\Community\Common7\Tools\

##### I can’t compile the project.

There’re two possibilities.

1. You need to download DirectX SDK. Search “DirectX Software Development Kit”.
2. If the problem is 'main': entrypoint not found. Exclude MultiAnimation.fx from build. (Right click MultiAnimation.fx -> Property -> Configuration -> General -> Excluded From Build. Select "Yes".)

##### Why there are two classes containing agent data?

The framework is taken from Microsoft DirectX 9 Demo. By default GameObject class contains all components of the agent.

Then in we added AgentBTData class that contains all data regarding the behavior tree system. We do this to minimize the change needed to made for this framework.

##### Then How to I get agent data?

To obtain agent’s GameObject instance. Do one of the following:

g\_database.Find(); // using id or agent name to find GameObject

AgentBTData::GetGameObject(); // get GameObjet from AgentBTData

To obtain agent’s AgentBTData instance. Do one of the following:

g\_trees.GetAgentData(“Agent Name”); // get AgentBTData for agent based on its name

g\_trees.GetAllAgentData(); // get all AgentBTData as array

##### How do I access other class instances?

The following are exposed to you:

g\_clock: clock class.

g\_database: game object container.

g\_random: custom random number generator.

g\_trees: behavior tree class.

##### What is the coordinate system for D3DXVECTOR3?

The range of value for x,y,z are [0, 1]. From the 2D plane, x is horizontal and z is vertical.

# Behavior Tree Framework

##### Why don’t we use unreal for this assignment?

1. There is no “standard” about how a behavior tree should work. Unreal has its own implementation, so does this framework.
2. Since all assignment basecode of this course are from the same DirectX 9 demo. Be familiar with this framework will help working on the next two assignments.

##### Is “Run 3 times” node considered as different node than “Run 5 Times” node?

Yes. This is to keep the framework simple and to make the project easier.

##### How do I create new node?

Depend on which kind of node you would like to create (control flow, decorator, leaf), find one appropriate node as template.

1. Create node files (.h and .cpp) base on existing node. Then change class name.
2. Then register new node in “BTResources\Nodes.def”.
3. Include node declaration in “BehaviorTreesDef.h”.
4. For each node, you must have definition for all virtual methods (OnInitial, OnEnter, OnUpdate, OnExit, OnSuspend).
5. You can create more methods and include a local blackboard for each node.

As reference, node examples are in “Source\BehaviorTrees\Nodes\folder”.

##### How do I create new tree?

1. First create node files.
2. Then run BehaviorTreeEditor.exe to create your behavior trees.
3. In world.cpp, register agent with the tree
4. Read course slides for more info.

##### How to create new agent?

Look at world.cpp line 90-106.

To register an agent with a particular behavior tree, do the following:

g\_trees.Register("agent name", "Tree Name");

g\_trees.GetAgentData(name).InitialTinyBlackBoard(npc); // or the initial method of your custom local blackboard

Warning: You have to initial a local blackboard for the agent.

##### What information stores in node logic, and what in node data?

Logic: node name, array of children logic

Data: pointer to node logic, pointer to agent, node status, array of children status, node blackboard

##### How to send message to other agents?

Use one of the following methods:

1. SendMsg(): send message to a particular agent.
2. SendMsgToAll(): send message to all agents.
3. SendMsgFromSystem(): send message to all agents from the system.

##### What are local blackboard?

There’re two: blackboard for an agent, blackboard for a node.

These are custom data instance that you are free to add new variables for the agent or a node.

Use GetLocalBlackBoard() to get the instance.

##### How do I set up a local blackboard for an agent?

Local blackboard instance is derived from AgentAbstractData class. If you wish to make a new one, you need to inherent from it also.

If you do not want to create your own custom class. By default, all agents in the framework uses TinyBlackBoard class as their local blackboard. You are free to modify this class to suit your need.

See InitialAgentAbstractData() and InitialTinyBlackBoard() to get an idea on how to initialize them.

##### How do I set up a local blackboard for a node?

Local blackboard instance is derived from NodeAbstractData class. If you wish to make a new one, you need to inherent from it also.

You need to also write your own InitialLocalBlackBoard() method, and execute it in OnInitial().

##### How to handle received message?

Incoming messages are stored in a message queue, and each frame it popped one message out. You handle them (as switch statement) in OnMessage() method of the local blackboard class you use for that agent.

##### How to reset a behavior tree of an agent?

Send MSG\_Reset message to the agent.

Beware. Agent can only handle reset request if its local blackboard handles such message (by default, both AgentAbstractData and TinyBlackBoard handle this message).

##### How to delete an agent?

Call AgentBTData::MarkForDeletion() method.

##### What are those virtual methods for every node?

OnInitial(): only run once when you start the program.

OnEnter(): only run once when the node is executed the first time after being reset (status is BT\_READY).

OnUpdate(): run every frame if the node is running, and OnEnter() has been executed.

OnExit(): only run once after the node returns “SUCCESS” or “FAILURE”.

OnSuspend(): run every frame if the node is marked as BT\_SUSPEND.

# Control Flow Nodes

##### What is the difference between composite node and parallel node?

Composite: runs one child at a time.

Parallel: runs multiple children at a time.

##### WHAT is the difference between composite node and parallel node, implementation wise?

RunChild() method is what makes all the difference.

unsigned RunChild(int child\_index, bool on\_same\_stack, bool stay\_stack, NodeData \*nodedata\_ptr);

Parameters:

child\_index: child index.

on\_same\_stack: flag: if child is pushed to the same stack.

stay\_stack: flag: if child stays on stack, after it returns SUCCESS or FAILURE.

nodedata\_ptr: current node data.

Think each tree as a stack. When you run a child from a composite node, you push that child on top of the stack. So each frame the framework only executes the top node of the stack.

For parallel node, you do not push children on the same stack. Think you have an array of stacks. And when you run a child, you add one empty stack and push the child onto that one. So each child of the parallel node is like its own sub-tree.

You use on\_same\_stack and stay\_stack flags to control the behavior. For a composite node, typically you set them to (true, false). And for parallel node, you set them to (false, true). You can also use other combinations to generate other types of behaviors.

##### What is the default EXIT condition for parallel node?

Unfortunately, there isn’t one. It means that you have to specify one if you want to create a parallel node.

Why?

For a composite node, it runs one child at a time, so it finishes either by meeting certain early exit condition, or runs all children. But for a parallel node, it runs all children at once, and after a child is finished, it records its return status then restarts again.

So, you would have to specify where you want the parallel node to finish, since there’s no way to “exit after runs all children”. Also since all children stay on their own stacks, when a parallel node finishes executing, it needs to manually call RemoveChildrenFromExecutionList() to get rid of all children.

See C\_ ParallelNode::OnUpdate() and ParallelNode::OnExit() for examples.

# Decorators

##### What is the difference between interrupter node and repeater node?

Interrupter: Every frame this node runs itself then runs its child.

Repeater: this node only runs after its child finishes executing, and returns “SUCCESS” or “FAILURE”. At this time, the child runs OnExit() once then stop executing.

##### My “run X times” node doesn’t work.

Because after a child of the given decorator returns “SUCCESS” or “FAILURE”, the child runs OnExit() the next frame then stop executing. In order to re-run the child, you need to call these manually:

ResetChildReturnStatus();

RunChild();

Note the first two boolean flags of RunChild() method.

on\_same\_stack flag: if it’s true, the decorator will not run unless its child finishes executing. True is for repeater, and False is for interrupter.

stay\_stack flag: if it’s true, the child will not stop executing even after it returns “SUCCESS” or “FAILURE”. If this flag is set to true, be sure to remove the child manually in OnExit() method.

# Leaf nodes

##### Why I can’t see any agent on screen?

This is the bizarre behavior from the DirectX demo.

The following two methods need to be called to set up tiny’s animation:

SetSpeedStatus();

SetTinySpeed();

Without them, the animation frame won’t be set, thus it won’t show any agent on screen.

The easiest way to avoid this problem is to always run Idle node first.

##### Why my agent moves weird?

Probably you run the above two methods in OnUpdate().

Since those two methods set the animation to the first frame. If you run them in OnUpdate, then agent will keep going back to the first frame of either walking or running.

These two methods only need to be executed once (usually in OnEnter()), unless you want to change the movement type for the agent.