I. Определение и описание дерева поведения.

Дерево поведения (Behavior Tree, BT) – ориентированный ациклический граф, узлами которого являются возможные варианты поведения ИИ.

«Ширина» дерева указывает на кол-во доступных действий, «длина» - на сложность.

Узлы BT – задачи, или поведения, имеют одно из четырёх состояний:

1. Успех – задача выполнена успешно

2. Неудача – условие не выполнено или задача невыполнима

3. В работе – задача запущена и ожидает завершения

4. Ошибка – в программе возникает неизвестная ошибка

Узлы BT можно разделить на две группы:

1. Неконечные (non-leaf nodes, рисунок 1). Должны иметь любое количество дочерних узлов (кроме корневого и декоратора, у которых должен быть только один дочерний узел). Неконечные узлы можно рассматривать как задачи, которые нужно выполнить. Каждая задача определяется поддеревом, состоящим из различных элементов. Существует два основных неконечных узла, которые дополняют друг друга, — это селекторы и последовательности. Также существуют декораторы, обеспечивающие большую функциональность ВТ, и параллельные узлы, добавляющие параллелизм.

2. Конечные (leaf nodes, рисунок 2). Определяют наблюдение и взаимодействие с игровой средой. Их называют действиями, условиями и ссылками. Конечные узлы часто рассматриваются как элементарные действия, которые должны быть как можно более краткими для поддержки возможности их повторного использования.

Выполнение ВТ происходит вглубь, начиная с корневого узла. Обычно все неконечные узлы будут выполнять их дочерние элементы слева направо, но есть исключения. Когда узел завершает выполнение, он возвращает статус, который может быть успешным, неудачным или исключением. Обстоятельства, при которых возвращается статус исключения, зависят от типа узла. Исключения возвращаются в том случае, если узел не смог вернуть ни успешного, ни неудачного статуса.

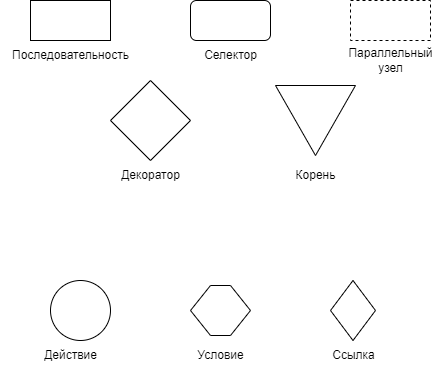


Рисунок 1. Неконечные узлы (non-leaf nodes) в деревьях поведения.

Неконечные узлы (рисунок 1):

1. Селекторы (Selector) – последовательно выполняет свои дочерние узлы слева направо, пока не получит успешный ответ, после чего сам селектор возвращает успешный ответ. Если же один из дочерних узлов вернёт неуспешный статус, будет выбран следующий дочерний элемент. Если все дочерние элементы вернут неуспешный статус, и узел селектора вернёт неуспешный статус.

2. Вероятностный селектор (Probability selector) – узел селектора с распределением вероятности по его дочерним элементам. Вероятность указывает, насколько возможен выбор дочернего элемента. Если выбранный дочерний узел вернёт неуспешный статус, распределение вероятности будет нормализовано по оставшимся дочерним узлам, после чего будет выбран новый дочерний узел. Возвращаемое значение вероятностного селектора выбирается по тем же правилам, что и для обычного.

3. Последовательность (Sequence) – узел последовательности будет выполнять каждый из своих дочерних узлов слева направо, пока все узлы не вернут успешный статус, либо пока один из них не вернёт неуспешный статус. В первом случае узел последовательности вернёт успешный, во втором случае – неуспешный статус.

4. Декоратор (Decorator) – добавляется в ВТ для обеспечения большей гибкости. Декоратор позволяет добавлять новое поведения без изменения существующего кода. Чаще всего используются в качестве фильтров, чтобы действия выполнялись с некоторым условием. Например, чтобы действие выполнялось определённое количество раз, с некоторой вероятностью и т. д. Помимо фильтрации декоратор можно использовать для практически любой модификации поведения. Декоратор может иметь только один дочерний узел.

Декоратор возвращает успешный статус, если заданные условия выполнены и дочерний узел был выполнен успешно.

5. Параллельный узел (Parallel node) - параллельно выполняет все дочерние элементы. Обстоятельства, при которых узел возвращает успех или неудачу, задаются разработчиком. Возможно, он будет аналогичен логике селектора или узла последовательности.

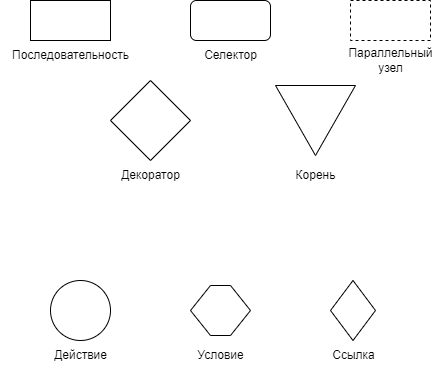


Рисунок 2. Конечные узлы (leaf nodes) в деревьях поведения.

Конечные узлы (рисунок 2):

1.. Условие (Condition) – наблюдают за состоянием игровой среды и возвращают успех или неудачу на основе наблюдения. Это может быть сравнение значений, проверка состояния и т. д.

2. Действие (Action) – используется для взаимодействия с игровой средой посредством контролируемых действий, таких как движение персонажей и взаимодействие с объектами. Возвращает успех при удачном выполнении действия, в противном случае – неудачу.

Действие должно быть как можно более простым, то есть представлять собой отдельные действия в игровом мире.

3. Ссылка (Link node) – содержит ссылку на корень другого ВТ. Выполняет узел связанного ВТ и ожидает ответа. Вводит модульность и возможность повторного использования поведений.

Для обмена информации между узлами и деревьями используется доска объявлений (Blackboard), хранящая информацию, полученную от узлов действия или условий во время выполнения. Хранение такой информации позволяет получить данные в некоторой части дерева и сохранить эти данные для дальнейшего использования.

Доска объявлений может быть общедоступной, чтобы любой узел мог считывать и записывать данные.

II. Математическая модель

Q = {q}i – множество состояний, q = <{p}i, {v}i>, где p – параметр состояния, v - его значение.

A = {a}i – множество действий, a: q => q' – функция, для состояния q определяющая новое состояние q' путём изменения значений {v}i некоторых параметров из {p}i (q = <{p}i, {v}i>, q' = <{p}i, {v'}i>).

C = {c}i – множество условий, c: <q, q'> => b – функция проверки соответствия состояния q состоянию q' путём сравнения значений некоторых параметров этих состояний, возвращает булево значение b {True, False}.

Разрабатываемая математическая модель должна с помощью определённых в ней множеств действий A и условий C совершить переход из начального состояния q0 в целевое состояние qn. Опишем такую модель, основанную на деревьях поведения.

M – модель дерева поведения, описывающая деревья для макро- и микроуправления.

N = {n}i – множество узлов, где n = <r, t, {n}i> - узел дерева поведения, где t {L, NL} – тип узла, указывающий, конечный узел (L) или нет (NL), а множество дочерних узлов {n}i пусто для конечных узлов и не пусто для неконечных.

{r}i – множество исполняемых функций r: QxN => S, определяющая для каждого состояния из Q и узла из N статус S {s, f, w, e}, где s – успех, f – неудача, w – в работе, e – ошибка.

Функция r в неконечных узлах запускает дочерние узлы, а в конечных либо выполняет действие a A, совершающее переход из одного состояния в другое, либо проверяет условие c C.

DTD-схема дерева поведения и его элементов может выглядеть следующим образом:

<!DOCTYPE behaviorTree[

<!ELEMENT behaviorTree (state, root)>

<!ATTLIST behaviorTree manager (macro|micro) "micro">

<!ELEMENT state EMPTY>

<!ATTLIST state

values CDATA #REQUIRED>

<!ELEMENT root (nonLeafNode)>

<!ATTLIST root id ID #REQUIRED>

<!ELEMENT nonLeafNode EMPTY>

<!ATTLIST nonLeafNode

id ID #REQUIRED

parentID IDREF #REQUIRED

childesID IDREFS #REQUIRED

status (noRun|success|failure|inWork|error) "noRun">

<!ELEMENT leafNode (action|condition|link)>

<!ATTLIST leafNode

id ID #REQUIRED

parentID ID #REQUIRED

status (noRun|success|failure|inWork|error) "noRun">

<!ELEMENT action EMPTY>

<!ATTLIST action

values CDATA #REQUIRED

newValues CDATA #REQUIRED>

<!ELEMENT condition EMPTY>

<!ATTLIST condition

values CDATA #REQUIRED

expectedValues CDATA #REQUIRED>

<!ELEMENT link EMPTY>

<!ATTLIST link

rootAnotherTree IDREF #REQUIRED>

]>

Каждое дерево поведения содержит в себе корень (root) – начальный узел, содержащий id дерева и один неконечный узел nonLeafNode, который содержит ссылку на родительский узел parentID, ссылки на один или несколько конечных или неконечных дочерних узлов childesID и статус выполнения status.

Каждый конечный узел leafNode также содержит собственный id и id родительского узла parentID, статус status и выполняемую функцию function, которая может быть трёх типов: action – изменяет значение параметров состояния values на newValues, condition – сравнивает значение параметров состояния values с expectedValues, link – содержит ссылку на корень другого дерева поведения rootAnotherTree.

Рассмотрим пример. Опишем с помощью разработанной DTD-схемы следующее простое дерево поведения, включающее в себя корень, узел последовательности и два конечных узла – условие и действие:

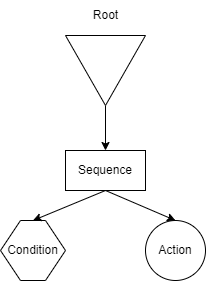


Рисунок 4. Простой пример дерева поведения.

XML-описание этого дерева в начальном состоянии согласно DTD-схеме выше будет выглядеть следующим образом:

<behaviorTree manager="macro">

<state firstValue="value1" secondValue="value2"></state>

<root id="#root">

<nonLeafNode id="#sequence" parentID="#root"

childesID="#condition #action" status="noRun">

</nonLeafNode>

</root>

</behaviorTree>

<leafNode id="#condition" parentID="#sequence" status="noRun">

<condition values="firstValue" expectedValues="value1">

</condition>

</leafNode>

<leafNode id="#action" parentID="#sequence" status="noRun">

<action values="firstValue secondValue" expectedValues="newValue1 newValue2"></action>

</leafNode>

Для упрощения данное дерево поведения будет оперировать только с двумя параметрами состояния – firstValue и secondValue.

При запуске дерева первым будет запущен узел #sequence, его статус сменится на «Run». #sequence запустит поочерёдно сначала конечный узел #condition, затем конечный узел #action.

Узел #condition проверит, равен ли параметр firstValue значению "value1". Так как условие выполняется, статус узла сменится на "success" и будет запущен следующий узел в последовательности childesID узла #sequence – узел #action.

Узел #action сменит параметры состояния firstValue и secondValue на значения newValue1 и newValue2 соответственно. Статус узла примет значение "success".

Так как все узлы из последовательности childesID узла #sequence приняли статус "success", то и сам узел примет этот статус.

Таким образом, произойдёт переход из состояния {firstValue: value1; secondValue: value2} в состояние {firstValue: newValue1; secondValue: newValue2}, и дерево поведения успешно завершит работу.

Модель в конечном состоянии будет выглядеть следующим образом:

<behaviorTree manager="macro">

<state firstValue="newValue1" secondValue="newValue2"></state>

<root id="#root">

<nonLeafNode id="#sequence" parentID="#root"

childesID="#condition #action" status="success">

</nonLeafNode>

</root>

</behaviorTree>

<leafNode id="#condition" parentID="#sequence" status="success">

<condition values="firstValue" expectedValues="value1">

</condition>

</leafNode>

<leafNode id="#action" parentID="#sequence" status="success">

<action values="firstValue secondValue" expectedValues="newValue1 newValue2"></action>

</leafNode>