Behavior Designer - Tactical Pack

Table of Contents

Behavior Designer - Tactical Pack

The Behavior Designer – Tactical Pack includes 13 tasks focused on tactical situations. The default set of Tactical Pack tasks use Unity's navigation mesh to traverse the world. The Tactical Pack doesn't do the actual movement – it instead sets the destination for the underlying pathfinding implementation (Unity's NavMesh, A* Pathfinding Project, etc).

Integrations

You can download the Tactical Pack integrations from this page. After you have imported an integration you can start to use the integration tasks by adding them to your behavior tree. For example, if you want to use the A* Pathfinding Project version of the attack task you would add the task located under Actions -> Tactical -> A* Pathfinding Project -> Attack.

Playmaker

The Tactical Pack tasks can use Playmaker to attack and receive damage. To have an agent attack using Playmaker, add the AttackBridge component to your agent. This component allows you to specify which Playmaker event should be triggered when an attack should take place. Optionally a Vector3 may also be specified which indicates the direction that the agent should attack.

Receiving damage with Playmaker is similar to attacking. Add the DamagableBridge component to any GameObject that can receive damage, and the Damage Event will be triggered when the GameObject should take damage. The Damage Amount and Health Variable Names are required. These variables map to a Playmaker variable which specify how much damage was received and the total health that the GameObject has left.

Interfaces

One of the initial design decisions that we had to make with the Tactical Pack was to define what it means to attack and take damage. Attacking and taking damage means different things to different games. For example, attack could mean to shoot a gun or throw a melee punch. Our goal with the Tactical Pack is to make the code as generic as possible. To solve this we added two interfaces, IAttackAgent and IDamagable. This allows you to define exactly what it means to attack or take damage for your game while still being able to use the Tactical Pack.

Included with the demo scene is one implementation of IAttackAgent and IDamagable. When the agent attacks they will instantiate a bullet prefab and that bullet will travel in the direction of the target. When the bullet hits the target it will call the IDamagable implementation to do the actual damage. It is expected that you will implement your own IAttackAgent and IDamagable components that fit your game. By structuring the Tactical Pack this way it succeeds in having a generic pack that works with any type of game.

The following methods need to be implemented with IAttackAgent:

```
// Returns the furthest distance that the agent is able to attack
from.
float AttackDistance();

// Can the agent attack?
bool CanAttack();

// Returns the maximum angle that the agent can attack from.
float AttackAngle();

// Does the actual attack.
void Attack(Vector3 targetPosition);
```

The following methods need to be implemented with IDamagable:

```
// Take damage by the specified amount.
void Damage(float amout);

// Is the object currently alive?
bool IsAlive();
```