Enemy Vision 2

How to use Enemy Vision for Unity

Enemy Vision 2

Enemy Vision is a simple and lightweight package for creating enemy behavior in a stealth type game. It's a great starting point if you are planning to implement that type of behavior. My goal is to keep this simple to use and easy to edit.

The best way to get started with the Enemy Vision system is to look at the demo scenes. The following document helps to explain more in details the different options available. There are two demos included: one 2D platformer, and one 3D top-down game.

NOTE: When importing the asset for the first time, depending of your Unity version the Layers name used in the demo may not be imported because unity doesn't package those with the asset. This will not break the demo since internally Unity uses layer numbers and not layer names. But if you wish to have the same layer names than me for better understanding of the demo set these layer names by clicking on "Layers->Edit layers": Layer 8 to "Obstacle" and Layer 9 to "Character".

Getting Started

Here are the main scripts available. Each of them has also a 2D version.

- Enemy.cs
- EnemyVision.cs
- EnemyAnim.cs
- VisionCone.cs
- VisionTarget.cs

Enemy

Basic enemy script handle enemy movement and the different enemy behavior (Patrol, Chase, Alert, Confused). Manages character movement and rotation. The 3D version is also compatible with Unity's navmesh pathfinding. This script will NOT transition automatically between state, and will always stay the state you set unless used with EnemyVision.

- **move_speed**: How fast the enemy moves
- **run_speed**: How fast the enemy moves while in Chase state.
- **rotate_speed**: How fast can it rotate
- **obstacle_mask**: Which layers are considered as "ground" or "obstacle that the enemy will try to avoid or stand on while walking. Can be set to "everything" by default.
- **fall_speed**: Speed when the enemy is falling
- **(2D) fall_accel**: Acceleration when the enemy is falling
- **(2D) ground_raycast_dist**: How far from the collider should it detects ground.
- **(3D) use_pathfind**: If on, will try to use unity's navmesh for pathfinding. Make sure the navmesh is setup correctly in your scene for this to work. If off, the enemy will bump into obstacles without trying to avoid them.
- **State**: active state of the enemy (Patrol, Chase, Alert or Confused). Can also be set to none for manual behavior (calling MoveTo yourself for example).

- **(3D) type**: Patrol pattern. Can be a loop, a back and forth line (rewind), or just facing without movement.
- **wait_time**: How long to wait when it reaches a "patrol point".
- **patrol_path**: List of positions the enemy will patrol to.
- **Alert_wait_time**: How long the enemy will wait before starting to walk in Alert state.
- **Alert_walk_time**: How long the enemy will walk in Alert state.
- **Follow_target**: the current target being followed (automatically set by EnemyVision)
- **memory_duration**: How long the enemy will guess the real position of the target after the target has been removed (for example after it lost sight of it).
- **target**: The target to follow

EnemyVision

Handles detection of the player. It creates a enemy vision display cone (VisionCone) and looks for the player (VisionTarget). It will swap between the different states in the Enemy script based on what is detected.

- **vision_angle:** How large is the enemy vision cone
- **vision_range:** How far can the enemy see
- **vision_near_range:** when within that range, the player will be detected immediately (skip the alert state)
- **vision_height_above:** Enemies can't see targets above them, except within this offset.
- **vision_height_below:** Enemies can't see targets below them, except within this offset.
- **touch_range:** In that range, the enemy will see 360 degree
- **vision_mask:** Layers that will block vision, make they are the same on the VisionCone object.
- **Group_detect:** when checked, will become alerted by other enemies if that enemy is already chasing the player (even if they didn't see the player directly).

- **Detect_time:** How long does it take to detect a target after it enters the vision cone
- **Alerted_time**: Duration the enemy will stay alerted when they notice something.
- **follow_time:** How long will the enemy chase a target?
- **no_return:** If on, enemy will not return to original patrol route after chasing a player. It will instead start looking around in random direction if the follow time expired.
- **eye**: reference position of the eye. Vision cone will start from this point and raycasts will start from this point.
- **vision_prefab**: the vision cone prefab

- **onSeeTarget**: Event triggered when the enemy sees a new target (Patrol->Alerted)
- **onDetectTarget:** Event triggered when it starts chasing the target (Alerted->Chase)
- **onTouchTarget:** Event triggered when the enemy reached the target

VisionCone

This must be attached to the enemy vision (red cone) instead of the enemy itself. EnemyVision will spawn a VisionCone for each enemy at the Start of a scene. This scripts will change the shape of the vision display based on obstacles in the scene. To help with optimization and frame rate, you can change the number of points that will create the cone (precision) and the refresh rate.

- **target**: The parent EnemyVision this vision cone belongs to.
- **vision_angle**: How large is the vision cone, set automatically by EnemyVision.cs
- **vision_range**: How long is the vision cone, set automatically by EnemyVision.cs
- **vision_near_range**: How long is the near part of the vision cone (only if using 2 levels), set automatically by EnemyVision.cs
- **obstacle_mask**: Layers that block the vision cone. NOT set automatically so make sure it matches the vision mask of the EnemyVision. (Because there are some rare cases where you will want the 2 to be different).
- **show_two_levels**: if checked, will show two different parts of the vision cone, far and near. Usually the near part will detect characters much faster than the far part.
- **Material**: which material to use for the cone
- **Material_far**: which material to use for the 2nd part of the cone in case that "show_two_levels" in on.
- **Sort_order**: renderer sort order.
- **precision:** How many vertex in the mesh to generate the vision cone. Reduce this to improve performance, increase it to have a more smooth cone.
- **refresh_rate:** Refresh rate in seconds of the cone, increase this to have better performance, reduce it or set to 0 to have the cone refresh more often (0 = each frame).

VisionTarget

Add this script to any object (like the player or a lure) that you want the EnemyVision to detect as a target. When this object is seen EnemyVision will change the state of Enemy.

- **visible**: When set to off, the target can't be seen. Useful for when hiding.

EnemyDemo

This script is specific to the demo characters, it just handles animations and events. It is also a good example of how to handle these for your custom character.

Improving the System

If you notice a bug or your think that a feature is missing, let me know about it. I REALLY WANT to improve this system and make it great! And since I can't predict all the use cases, your feedback would really help me know what I should include in the future versions.

Keep in mind that the goal is to keep the package lightweight for easy modification. I'm totally up for adding new features as long as it keeps the amount of code low. It's important to me that this system is easy to use and edit.

If you have any questions or suggestions send me an email:
contact@indiemarc.com

Thank you!

Credits

Indie Marc (Marc-Antoine Desbiens)
Freelance Game Developer
https://indiemarc.com