EASY AI SYSTEM

Basic manual

Overview

This system allows you to easily implement quite complex AI (artificial intelligence) behavior for any enemies or allies in your game. Path finding functionality included (allows avoiding even dynamic objects/obstacles)!

The system is extremely easy to setup and tune - just in several clicks you'll able to implement it anywhere. It really fast and performance safe (you can adjust it to get better accuracy or even higher performance). The system doesn't require any specific actions or updates for your objects and scenes.

List of predefined actions (you can easily change/add them if needed): Idle, Seek, Follow, Move, Attack, RunAway, Patrol, Heal, Die

This system works on all platforms supported by Unity3D.

How to use

To use this system – you should just:

1st option (auto):

1. Add(drag and drop) all prefabs from "_Easy_PathFinding_System_" to Hierarchy window
2. Add child object (with animation) to this object and assign it to *AnimatedObject* property of "AI_Controller" component.
3. Tune parameters in "AI_Tuner" component.

2nd option (manual):

1. Create game object (preferable with Rigidbody and Collider) and assign *PathFinding*, *PathFollowing* and *AI_Controller* components (scripts) to it
2. Create child game object (with Rigidbody and Collider) and assign *AI_Sensor* component to it
3. Assign this object(with *AI_Sensor*) to **viewTrigger** property of AI_*Controller* component
4. Assign *PathFinding* script to **pathFindingScript** property of *PathFollowing* component
5. Assign *PathFollowing* script to **pathFollower** property of AI_*Controller* component
6. Add child object (with animation) to this object and assign it to **AnimatedObject** property of "AI_Controller" component.
7. Add/setup needed actions
8. Write *enemyTag* and/or specify *FollowAlly* properties in "AI_Controller" component.

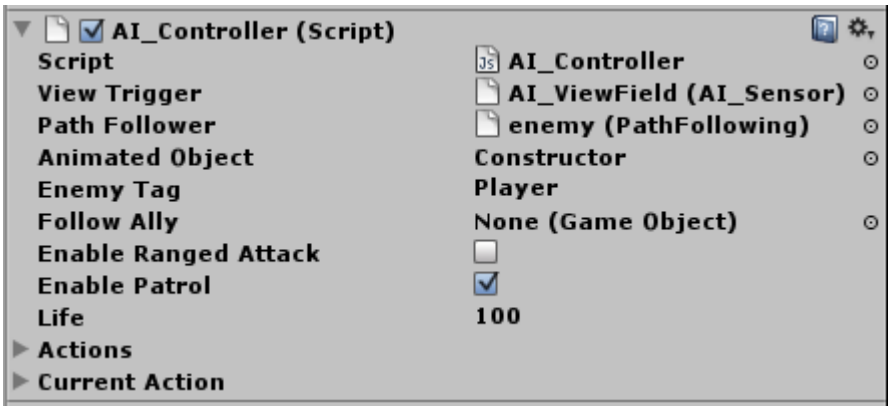
Please don't hesitate to contact me in any reason by mail: ***AllebiGames@gmail.com***

BASIC SCRIPTS DESCRIPTION

AI_Controller script description

Main script, that controls/manage all AI states and requests related actions (and animations).
Script also conected to PathFinding system, it handles navigation activity.

Average structure looks like:



- **viewTrigger** - Script attached to object used as view-field of AI. It's collider size useful for ranged attack distances too
- **pathFollower**- Script to process PathFinding and PathFollowing
- **animatedObject** - Object that contains visual player appearance and animations
- **enemyTag** - Tag of object, that regarded as enemy
- **followAlly** - If specified - AI will follow this object/character
- **enableRangedAttack** - Enable range attack. Only mele is allowed if false
- **enablePatrol** - If true - AI will patrol territory instead of simple Seeking
- **life** - Initial/max amount of life
- **actions** - List of all possible AI actions.
    - Basic and universal parameters for actions specified in class AI_Action
    - **Caption** - Just a caption for more comfortable navigation
    - **actionType** - Action type. Function with this name will be called to perform this action
    - **actionPriority** - Priority. Only actions with higher priority can interupt actions with lower
    - **actionDuration** - During this time action can't be interupted by action with lower priority.
    - **animationClip** - Play this animation during the action
    - **loopAnimation** - Specifies should be animation looped during action Duration or not
    - **customValue** - Just some value that you can use in functions to setup/tune some parameters of
- **currentAction** - Current requested action - this is internal variable, but it should be public. Please don't change it blindly

AI_Sensor script description

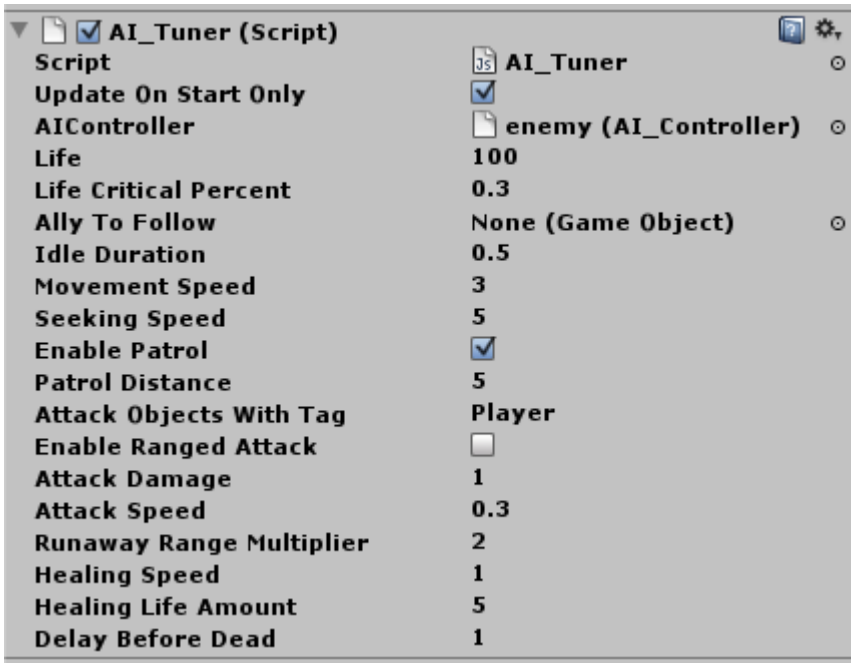
This script checks and saves tagged object collided with this one. It allows getting tag of triggeredObject
Script should be attached to object with Rigidbody and Collider (IsTrigger = true) attached

- **triggeredObject** - Object collided with this one

AI_Tuner script description

Simple help script allowing to tune AI parameters in attached AI_Controller easier.
This script is suitable for current (template) realisation of AI_Controller. If you will change AI_Controller - please update this script accordingly

Average structure looks like:

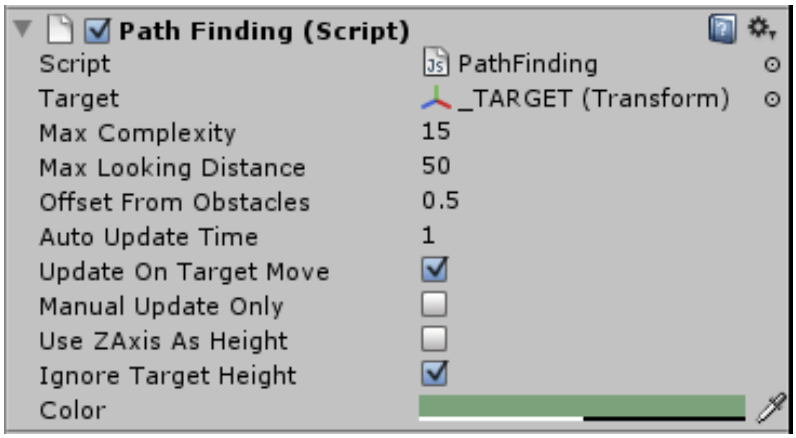


- **updateOnStartOnly** - Allow to specify should parameters be updated OnStartOnly or every frame
- **AIController** - Link to main AI script to tune
- **life** - Initial/max amount of life
- **lifeCriticalPercent** - How less life is critical (AI will try to runaway for healing)
- **allyToFollow** - If specified - AI will follow this object/character
- **idleDuration** - How long should Idle-state be
- **movementSpeed** - Speed of movement
- **seekingSpeed** - For Seeking-state - how fast AI will look around
- **enablePatrol** - If true - AI will patrol territory instead of simple Seeking
- **patrolDistance** - Max distance of movement in one patrolling step
- **attackObjectsWithTag** - Tag of object, that regarded as enemy
- **enableRangedAttack** - Enable range attack. Only mele is allowed if false
- **attackDamage** - Damage per one attack
- **attackSpeed** - How often will AI try to hit enemy
- **runawayRangeMultiplier** - Multiplier that allows to increase max distance oto run away from enemy
- **healingSpeed** - How fast will AI use healing ability if damaged
- **healingLifeAmount** - How much healt should be restored at once
- **delayBeforeDead** - How much time AI should wait before destroying itself (useful for death animation)

PathFinding script description

Main script of this Path finding system. Calculates (find) path automatically or according to specified rules Generates array of waypoints (around obstacles) until target will be reached

Average structure looks like:

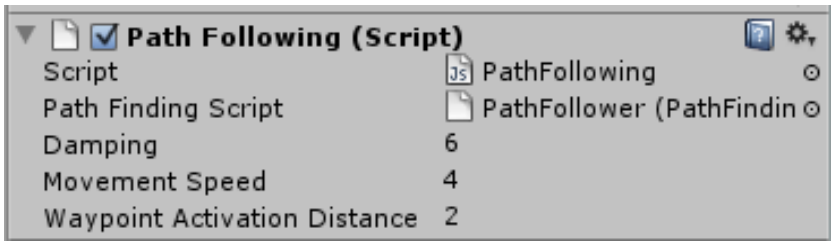


- **target** - Target/final point to build path to
- **waypoints** - Array of generated waypoints
- **maxComplexity** - Max number of waypoins in path
- **maxLookingDistance** - Max distance of raycasts (looking for waypoint)
- **offsetFromObstacles** - Set additional offset between waypoints and obstacles
- **autoUpdateTime** - Delay to next path recalculation. Works automatically if updateOnTargetMove and manualUpdateOnly = false;
- **updateOnTargetMove** - Update only if new target position different from previous one
- **manualUpdateOnly** - Allow only manual updates by calling "FindPath" function
- **useZAxisAsHeight** - By default path calculates in XZ plane, set it to true to use XY plane
- **ignoreTargetHeight** - Ignore target Y (or Z) offset from this object
- **color** - Debug path-visualization color

PathFollowing script description

This is example script to follow path.
It manages waypointed path from pathFinding Script and move object along it.

Average structure looks like:



- **pathFindingScript** - Path holder/generator script
- **damping** - Smooth facing/movement value
- **movementSpeed** - Speed of object movement along the path
- **waypointActivationDistance** - How far should object be to waypoint for its activation and choosing new
- **stuckDistance** - Max distance of move per regenTimeout that supposed to indicate stuck
- **stuckTimeout** - How fast should path be regenerated if player is stuck