Changelog / Release notes for RaptorAI

Note that some PDF readers allow you to click on the various sections in the table of contents, while others do not.

Table of Contents

RaptorAI 2 (RAMPA) 2

- Version 2.2.1 2
- Version 2.2 2
- Version 2.1.1 3
- Version 2.1 5
- Version 2.0.1 6
- Version 2.0 6

RaptorAI 1 7

- Version 1.3.1 7
- Version 1.3 7
- Version 1.2 8
- Version 1.1 8

RaptorAI 2 (RAMPA)

Version 2.2.1

- Editor Improvements
  - Fixed editor freezing for several seconds when there are a lot of assets in any resource folder
  - There are now caches for behaviour trees, animation profiles and groups

Version 2.2

- AIManager updates
  - ✓ New sense / object detection system with multi-threading
    - Detection is now handled by DetectionUpdate function in AISenses
    - All objects sent to DetectionUpdate in senses are now guaranteed to be proper type so normal casting can now be used
- New waypoint system
  - ✓ Waypoint updating is no longer done through the "NavigateWaypointPath" function call, but now runs on its own. Use "reachedEndOfPath" instead to check if the agent is at the end of the path
  - ✓ "navigatingPath" bool that indicates whether or not the system should call the "NavigateWaypointPath" function
  - ✓ "currentWaypoint" can now be set
  - ✓ More calculation methods in AIWaypointPath that let's you calculate how far to waypoints and other things like that
  - ✓ Advanced waypoint systems now support caching of routes
    - Option to cache all routes on awake
    - Can be more than 1000x faster than calculating the route each time
- New AI Modules
  - ✓ Modules run constantly and can be turned on / off by states for instance
  - ✓ Run concurrently with states and multiple modules can run at the same time
  - ✓ New Auto slope module that rotates agent to fit terrain
    - Able to extract terrain data directly or use raycasting to calculate normal
- New AIFastCameraSense and object that is faster than using the visual sense
  - ✓ Supports line-of-sight
- New Universal Memory. Useful for auto slope module if you use prefabs as manual assignment of terrain doesn't work for prefabs
  - ✓ New AIUniversalMemoryAdder component that let's you automatically add objects to the universal memory on awake
- New group system
  - ✓ You can now create groups and define relationships towards other groups

- Very basic as of now, but will be developed further in future updates
- Rewritten AIDatabase and AIMemory classes for better performance
  - ✓ No need to specify type when adding / setting items
  - ✓ More functions
    - Some of them return ItemList or ItemList<Cast> (wrapper lists) instead of creating new arrays all the time. Use them like a normal array.
- Other changes
  - ✓ Custom DestroyAgent method in AIController to properly destroy agents (use this instead of the normal Destroy function)
  - ✓ AIUniversalMemoryAdder and AIObjectManager now appear in the component menu under “RaptorAI”
  - ✓ Upgraded support to A* Pathfinding Project 4.2.5
  - ✓ Warning in Controller when having duplicate states / modules
  - ✓ Debugging changes to work with new AIDatabase and AIMemory
  - ✓ Added “reachedEndOfPath” variable in AINavigator that checks specifically if reached end of waypoint path
  - ✓ New IAgentAttachable interface used for assigning AIController to the various components and provides a “OnDestroyAgent” function for proper cleanup when destroying agent
  - ✓ New IdleNavigator that doesn’t perform any navigator functions and is used for agents that simply don’t need any form of movement
  - ✓ Settings are no longer overridden when upgrading project
- Other fixes
  - ✓ Some errors now give more information about what went wrong
  - ✓ Fixed AIAstarNavigator recalculating paths multiple times when reaching destination
  - ✓ Fixed errors when editing / viewing prefab with AIController
  - ✓ Other bugfixes
- A lot of small adjustments here and there

Version 2.1.1

- AINavigator changes
  - ✓ "currentWaypoint" can now be accessed (integer describing current waypoint index)
  - ✓ New "remainingDistanceSquared" function in AINavigator that is faster than using "remainingDistance". This is faster for comparing two distances for example. Remember that the value you compare it to has to be squared as well
  - ✓ "aiController" variable in AINavigator can only be set within the class (subclasses can still access it)

- ✓ "aiWaypointPath" variable that replaces SetWaypointPath and GetWaypointPath
    - ✓ New "currentDistanceFromLastWaypoint" variable that gives you the distance from last waypoint to the current position on the line connecting this waypoint and the next. If the AI agent is perfectly on the visible line between the waypoints, it will simply use the AI agent's position. If not, it will use the point that creates a 90-degree angle between the visible line and the AI agent's position. It might sound confusing at first, but it is very useful if you want to find the point where the AI agent would have been if it wasn't off-course. See "currentWaypointPathPosition" below for more on this.
    - ✓ New "currentWaypointPathDirection" variable that gives you the normalized direction from the last waypoint to the current waypoint. When following a waypoint path, this is the direction the AI agent should be heading at any given time
    - ✓ New "currentWaypointPathPosition" variable that gives you a point on the line between the last waypoint and current waypoint that best corresponds to the AI agent's position. If the AI agent is exactly on this line / path, it will give you the AI agent's position. If not, it will give you the point that creates a 90-degree angle between the line / path and the AI agent's position. The "currentDistanceFromLastWaypoint" is the distance between this point and the last waypoint.
    - ✓ New "progressTowardsNextWaypoint" variable that returns a percentage (value between 0 and 1) that tells you the progress from the last waypoint to the current waypoint. It is the same as the "currentDistanceFromLastWaypoint" over the distance between the last waypoint and current waypoint. This means that it calculates this not using the AI agent's actual position, but it's closest position compared to the actual line / path
- DropDown improvements
    - ✓ Ability to change between using AssemblyQualifiedName and FullName when using DropDown attributes (first one is used when using System.GetType)
- Other changes
    - ✓ Obsolete functions are now fully removed and can no longer be used
    - ✓ New behaviour configuration keyboard shortcut [CTRL] + Q → Displays context menu
    - ✓ There is no longer an error when destroying gameobjects containing the AIController script
    - ✓ AI Controller Editor optimization
    - ✓ Fixed tracing issue when trying to trace memory values using the AI Inspector

Version 2.1

- Senses and objects now have access to the AI Controller (applies to custom senses / objects)
  - ✓ Get access through the “aiController” variable
- AINavigator updates
  - ✓ “GetRequiredComponentTypes” now works for navigators and adds the required components to the top parent automatically. Use this instead of the [RequireComponent] attribute
  - ✓ Transition from regular functions to variables with getters and setters. For example, “SetDestination” has now become just “destination”. The functions have now become deprecated and existing navigator scripts should be updated
  - ✓ “remainingDistance” and “reachedDestination” have now become optional to override due to new default code.
  - ✓ “GetSpeed” has been separated into “currentSpeed” and “maxSpeed”
  - ✓ New currentVelocity variable that gives you the current speed + direction. “currentSpeed” is the magnitude of this.
  - ✓ New “stoppingDistance” variable
- New AIGenericNavigator class for faster creation of new navigators
  - ✓ Displays “Max Speed” and “Stopping Distance” automatically by default in AI Controller
  - ✓ Automatically initializes the rest of the variables so that you only have to override the “UpdateNavigator” function and write the movement code
  - ✓ Use it by making your navigator a subclass of AIGenericNavigator instead of AINavigator
- Changes to animation profiles
  - ✓ You can now use a new animation set as a fallback action
  - ✓ Int values are now supported
- Changes to Settings
  - ✓ You can now choose if you want unused scripts (states / navigators) to automatically be removed
  - ✓ You can now toggle visibility of the “RaptorAI Manager” GameObject in the hierarchy. There is also a button to show it in the inspector if you choose to have it hidden in hierarchy. This option is on by default.
- Changes to behaviour tree
  - ✓ Ability to move nodes up / down layer in addition to moving up / down index
  - ✓ Ability to replace nodes by right-clicking them (without deleting its children)
  - ✓ Keyboard shortcuts. These apply if your cursor is over a node
    - ▪ [Del] → Displays delete prompt

        - [Ctrl] + [Del] → Deletes node without displaying warning
        - [Ctrl] + [Arrow Up] → Moves node up one index
        - [Ctrl] + [Arrow Down] → Moves node down one index
        - [Ctrl] + [Arrow Left] → Moves node up one layer
        - [Ctrl] + [Arrow Right] → Moves node down one layer
        - NOTE: You can use right ctrl as well as left ctrl
- Other changes
    - ✓ You can now use various attributes in your states to show a dropdown menu of the available navigators / states / senses / objects in the AIController instead of regular string input
    - ✓ New generic dropdown attribute [DropDown] where you select yourself what should be in the dropdown list in the AIController
    - ✓ Animator variable is no longer hidden if no animation profile is selected
    - ✓ AIBehaviour renamed to AIBehaviourNode

Version 2.0.1

- New examples
    - ✓ Two new example scripts to demonstrate how you create your own navigators and nodes
- New navigator movement update function
    - ✓ Navigators now have a dedicated update function for handling the actual movement. This is updated along with the AIController
- Minor additions
    - ✓ Added missing summaries
    - ✓ Added more comments

Version 2.0

- Behaviour trees for visual scripting
    - ✓ Create AI behaviour without code using the AI behaviour tree configuration panel
    - ✓ Easy debugging
        - Ability to replay ticks to debug in-depth
        - Descriptive error messages when something goes wrong
        - Debug node that can be used
    - ✓ Choose from a range of behaviour nodes in your tree
    - ✓ Able to use values from memory and database
    - ✓ Possible to switch behaviour tree on runtime
        - Possible to switch to and from a state in addition to this
    - ✓ Easily create your own nodes to perform custom actions
    - ✓ Help button to quickly describe what each node does

    - ✓ Tooltips for node properties to tell you what they are used for
- More flexible memory and database retrieval methods
    - ✓ Getting values are no longer dependant on having exact type
    - ✓ Casting is now possible when retrieving values
- AI Inspector
    - ✓ Allows you to inspect memory and database values in real time
- Navigator updates
    - ✓ A* Pathfinding Pro is now supported by using the new navigator
    - ✓ More functions added to all navigators
- Improved UI
    - ✓ Everything except senses and objects is now configured with the AI Controller script attached to the AI
- New scent sense and scent object
    - ✓ Allows you to create AI that can "smell" moving or stationary objects
    - ✓ Scent objects leaves behind a trail that can be tracked and followed by using the scent sense
- Re-coded collision system
    - ✓ 100% now utilizes unity's built-in collision system
- Other changes
    - ✓ Small raptor icon next to scripts ☺

RaptorAI 1

Version 1.3.1

- Unity 5.6 upgrade
- Access AI Navigator, Memory, Local and Global Database, Start State and Tag using getter variables instead of functions e.g. aiNavigator instead of GetAINavigator(). The old function methods are still available

Version 1.3

- New collision system
    - ✓ Now only two Visual Types in AIVisualSense
        - Camera
        - Collider
    - ✓ Spheres and capsules are no longer available to use as collision as they massively impacted performance

    - ✓ Custom box collider (again for performance)
    - ✓ High accuracy support for custom box collider
    - ✓ You can make it less accurate as 100% accuracy would not be realistic for a person for instance
- Fixed UI:
    - ✓ Full multi-editing support of objects
    - ✓ Custom editors for all engine components (except AINavmeshNavigator)
    - ✓ Only show certain variables if certain conditions are met
- Other changes:
    - ✓ Banner on objects are now half the size
    - ✓ Changed some names, namely enum names
    - ✓ Fixed example scenes due to changes
    - ✓ Added minimap camera in maze example scene
    - ✓ Changelog file included in asset package

Version 1.2

- Updated AI Settings
    - ✓ Tag system to easily differantiate between different AI types in same state
    - ✓ Compare tags by using the AITag.SameAs function (not case sensitive) or the AITag.SameAsRaw (case sensitive). You can therefore choose yourself if you want your tags to be case sensitive or not.
- Global Settings
    - ✓ Change settings for all AI's
    - ✓ Change settings for all AI's with a certain tag - All with one button (this will be available under the RaptorAI menu -> Create Global Settings Object)
- Performance boosts
    - ✓ Able to modify how often different states should be updated
    - ✓ The UpdateState function from the AIState class no longer returns a float. If you want to delay the next update, use the DelayNextUpdate function instead. This will stack with the AI Settings call delay and if you call this function more than once, the effect will also stack.
- Other changes
    - ✓ Updated UI + logo banner to easily recognize AI components
    - ✓ Another new example scene demonstrating a maze setup

Version 1.1

- Few minor changes
    - ✓ Fixed proximity collider gizmo draw location
    - ✓ Added new example scene / scripts demonstrating animal AI