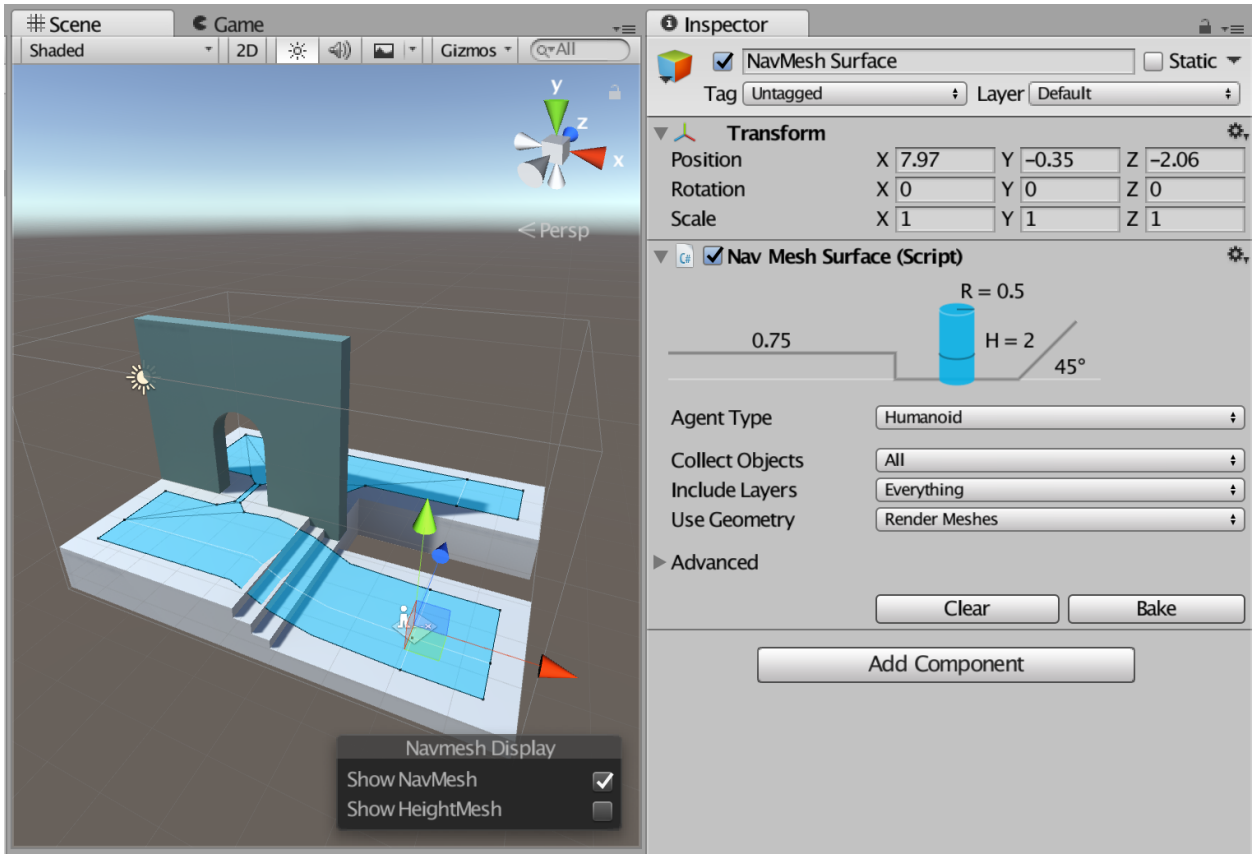
High-level NavMesh Building Components

Here we introduce four high level components for the navigation system:

- **NavMeshSurface** – for building and enabling a navmesh surface for one agent type.
- **NavMeshModifier** – affects the navmesh generation of navmesh area types, based on the transform hierarchy.
- **NavMeshModifierVolume** – affects the navmesh generation of navmesh area types, based on volume.
- **NavMeshLink** – connects same or different navmesh surfaces for one agent type.

These components comprise the high level controls for building and using NavMeshes at runtime as well as edit time.

NavMeshSurface



The NavMeshSurface component represents the walkable area for a specific agent type. The NavMesh Surface component defines a part of the world where a NavMesh should be built. A scene can contain multiple NavMesh Surfaces.

The preferred way to use the NavMeshSurface component is to create an empty Game Object containing the NavMeshSurface component. There's a menu option for that: *GameObject* > *AI* > *NavMesh Surface*. This creates an empty Game Object with a NavMeshSurface component attached to it.

The NavMeshSurface component can be also added to any game object. This is useful for cases where you want to use the Hierarchy to define which objects contribute to the NavMesh.

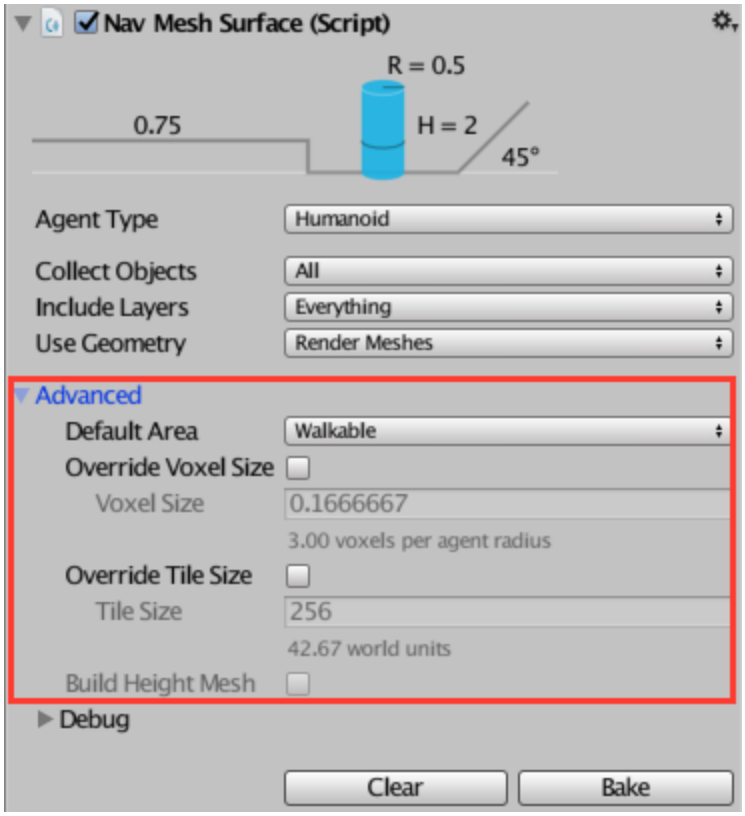
Parameters

- **Agent Type** – the agent type which will use this NavMesh Surface. The agent type is used for bake settings, as well as to match a NavMeshAgent to proper surface during pathfinding.
- **Collect Objects** – defines on a high level which objects should be used for baking.
    - *All* – use all active objects.
    - *Volume* – use all active objects overlapping the bounding volume (defined later)
    - *Children* – use all active objects which are children to the NavMeshSurface component, in addition to the object the component is placed on.
- **Include Layers** – defines the layers on which the objects must be to be included in the bake. This allows further culling of objects from inclusion in the bake e.g. effects or animated characters.
- **Use Geometry** – selects which geometry is used for baking.
    - *Render Meshes* – use geometry from rendered meshes and terrains
    - *Physics Colliders* – use geometry from colliders and terrains. When using physics this is usually a better option than *Render Meshes*. This way the agents will be closer to the physical bounds of the environment.

The main settings for the NavMesh Surface component allow you to filter the input geometry on a broad scale. To fine tune how input geometry is treated on a per-object level when building a navmesh, see NavMeshModifer component.

Game Objects which have a *NavMesh Agent* or *NavMesh Obstacle* will be excluded from the baking process automatically. They are dynamic users of the navmesh – and hence should not contribute to the navmesh building.

Advanced Settings

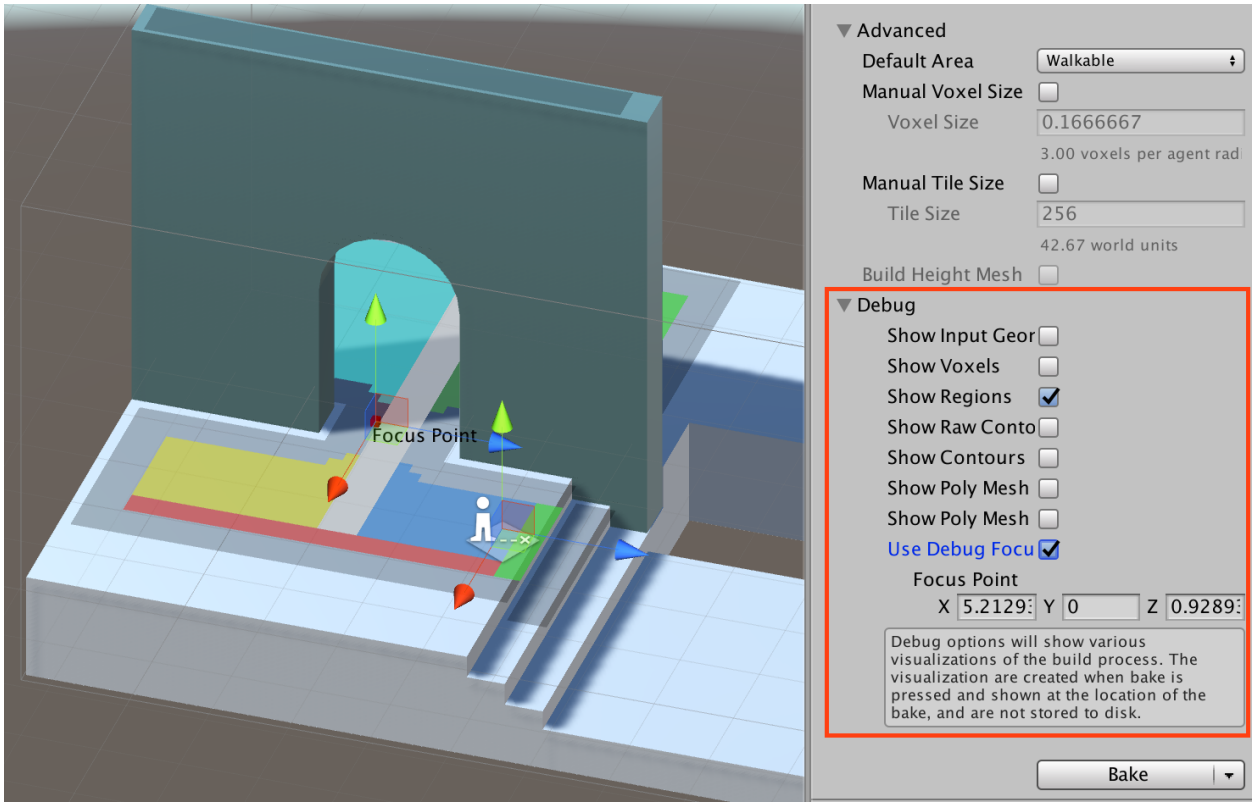


The main settings will cater for most use cases, but for anything not covered by the main settings, the advanced section has the following additional parameters

- **Default Area** – defines the area type generated when building the navmesh. The default value is *Walkable*. The NavMeshModifer component can be used to modify the area type in more detail.
- **Override Voxel Size** – The override voxel size controls how accurately the input geometry is processed for NavMesh baking. It is a tradeoff between speed and accuracy. A good value to start with is 3 voxels per agent radius (6 per diameter). This allows to capture most of the narrow passages, like doors and still have quick baking. If you have big open areas, you might go down to 1 or 2 to speed things up. Or if you have tight indoor spots, you can use smaller voxels, and use maybe 4-6 voxels per radius. More than 8 is usually not really worth it.
- **Override Tile Size** – In order to make the bake process parallel and memory efficient, the world is divided into tiles for baking. The white lines you can see on your NavMesh are tile boundaries. The default tiles size is 256 voxels. It is a good trade-off between memory usage and NavMesh fragmentation. The smaller the tiles are the more fragmented the NavMesh is, and this can sometimes cause non-optimal paths. NavMesh carving also operates on tiles. If you have a lot of obstacles, you can potentially speed up carving by making the tile size smaller, say 64-128. Also, if you plan to bake the NavMesh at runtime, you may choose to use smaller tile size to keep the maximum memory usage low.

- **Build Height Mesh** – Not supported yet.

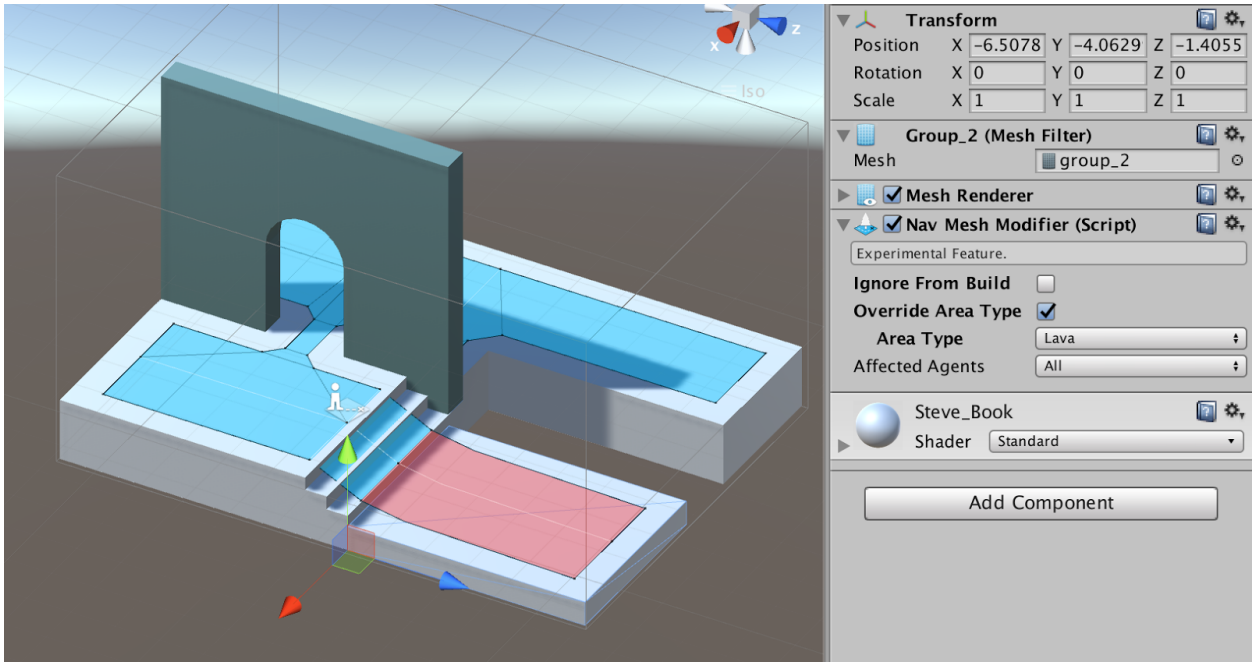
~~Advanced Debug Visualization [Not yet available]~~



~~The debug visualization is not generally needed, but when things go south, this can be a valuable tool to let us help you to find the culprit.~~

~~The different options show each step of the NavMesh building process from input scene voxelization to region splitting, contour generation and finally the NavMesh polygons. Since the process can generate a lot of data, some of which overlapping, the *Focus Points* helps you to narrow down the visualization for just one tile.~~

NavMesh Modifier



NavMesh Modifier allows to fine tune how a specific object behaves during NavMesh baking. In the above picture, the lower platform has modifier attached to it, which sets the object to have Lava area type.

The NavMesh Modifier affects hierarchically, that is, the Game Object where the Components is attached and all of its' children are affected. If another NavMesh Modifier is found further down the transform hierarchy it will override the modification for its children.

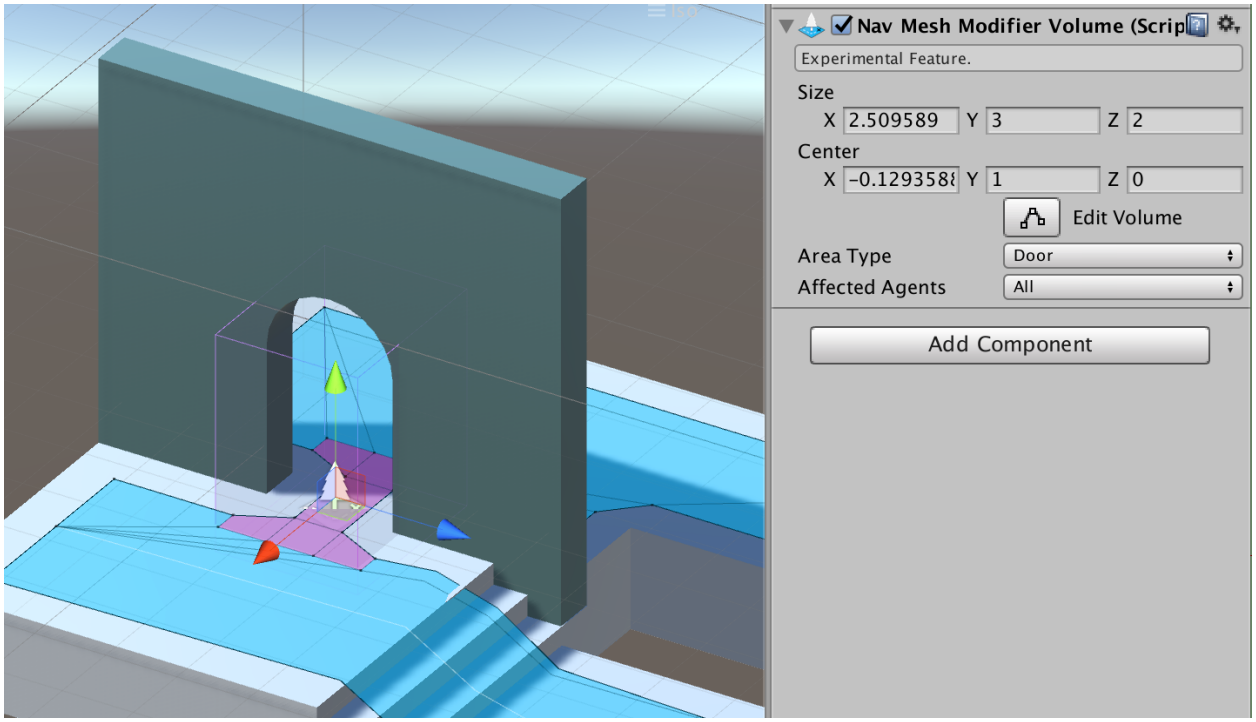
The NavMesh Modifier affects the NavMesh generation process, this means the NavMesh has to be updated to reflect changes to NavMesh Modifiers.

Note: This component is a replacement for the old setting which could be enabled from the Navigation window Objects tab as well as the static flags dropdown on the GameObject. This component is available for baking at runtime, whereas the static flags are available in the editor only.

Parameters

- **Ignore From Build** – when checked, the object and all if its' children are skipped from the build process.
- **Override Area Type** – when checked the area type will be overridden for the game object containing the Modifier and all of it's children.
    - **Area Type** – new area type to apply
- **Affected Agents** – a selection of agents the Modifier affects. For example, you may choose to exclude certain obstacles from specific agent.

NavMesh Modifier Volume



NavMesh Modifier Volume allows you to mark the area that falls inside the volume with specific area type. Where NavMesh Modifier marks certain objects with an area type, the Modifier Volume allows change the area type even more locally based on a volume.

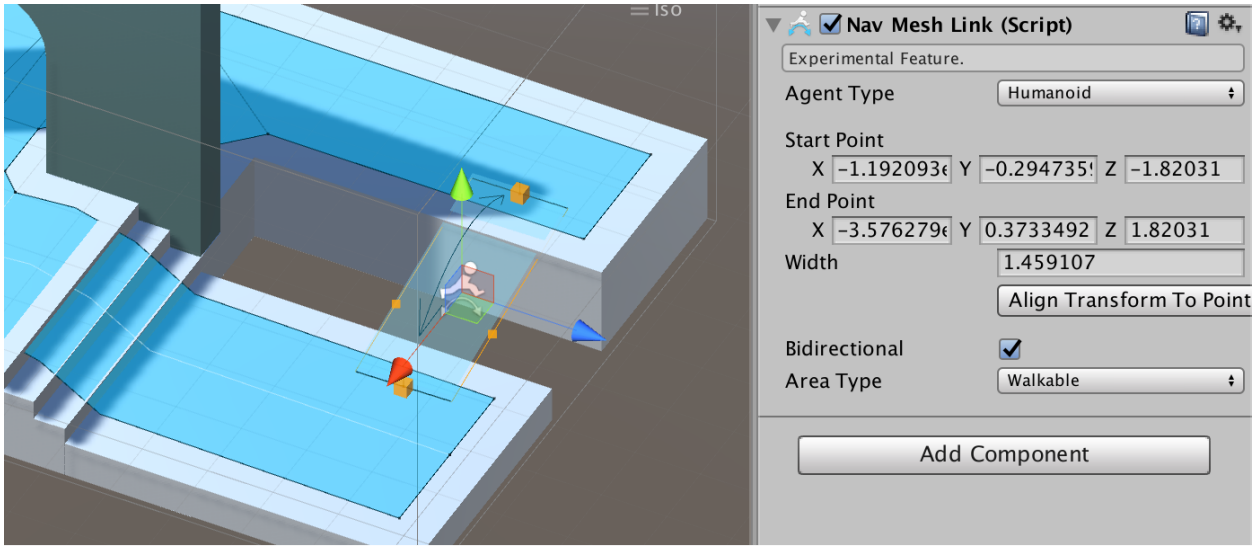
The modifier is useful for annotating certain areas over walkable surfaces which might not be represented as separate geometry, e.g. danger areas. It can be even be used to make certain areas non-walkable.

The NavMesh Modifier Volume affects the NavMesh generation process, this means the NavMesh has to be updated to reflect changes to NavMesh Modifier Volumes.

Parameters

- **Size** – dimensions of the modifier volume.
- **Center** – center of the modifier volume relative to the GameObject center.
- **Area Type** – describes the area type which the volume applies.
- **Affected Agents** – a selection of agents the Modifier affects. For example, you may choose to create danger zone for specific agent type only.

NavMesh Link



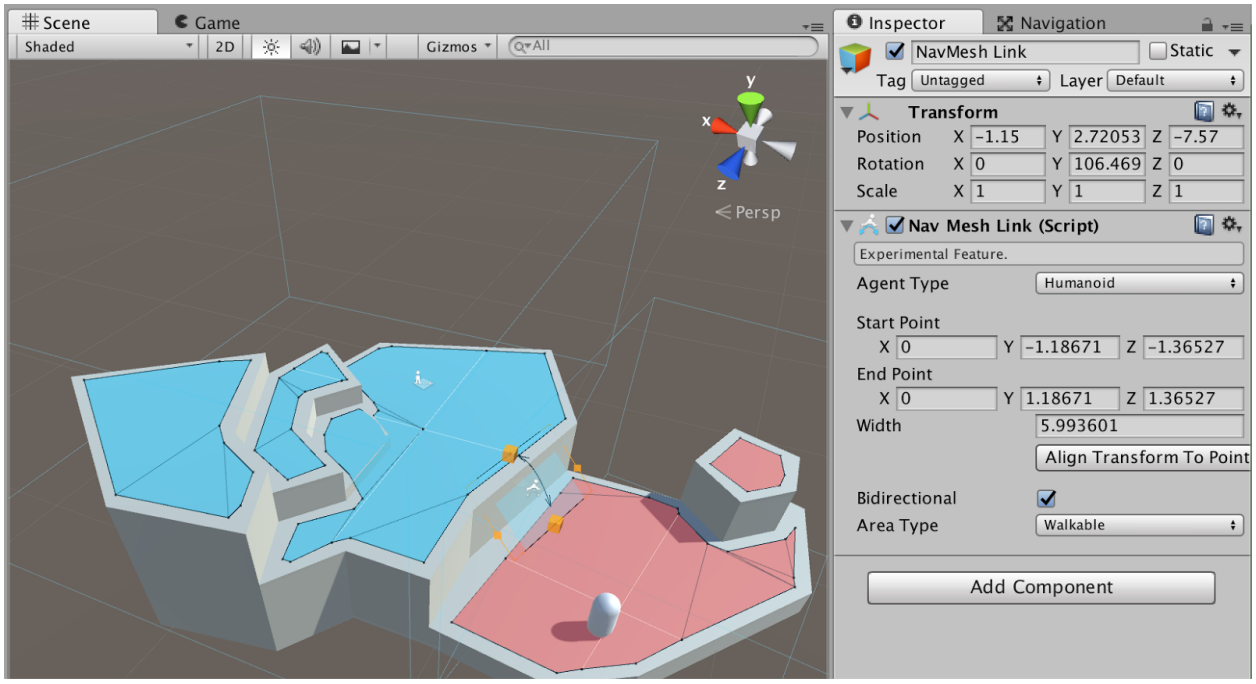
NavMesh Link allows to create a navigable link between two locations. The link can be from point-to-point, or it can be wider in which case the agent uses the nearest location along entry edge to cross the link.

The link is necessary to connect different NavMesh Surfaces

- **Agent Type** – the agent type which can use the link.
- **Start Point** – start point of the link, relative to the Game Object.
- **End Point** – end point of the link, relative to the Game Object.
- **Align Transform To Points** – clicking this button will move the Game Object at the links center point and alight the transform's forward axis towards the end point.
- **Cost Modifier** – When the cost modifier value is non-negative the cost of moving over the NavMeshLink is equivalent to the cost modifier value times the Euclidean distance between NavMeshLink end points.
- **Bidirectional** – when checked the link can be traversed from *start-to-end and end-to-start*, when unchecked only from *start-to-end*.
- **Area Type** – the area type of the link (affects path finding cost)

Techniques

Connecting Multiple NavMesh Surfaces Together



If it is desired to allow an agent to move along multiple NavMesh Surfaces in a Scene, the surfaces need to be connected together using NavMesh Links.

In the example scene above, the blue and red NavMeshes are defined in different NavMesh Surfaces. A wide NavMesh Link is added, it spans from one surface to another.

Things to keep in mind when connecting surfaces:

- You can connect surfaces using multiple links.
- Both the surfaces and the link must have same agent type.
- The link's start and end point must be only on one surface. It is OK to have multiple NavMeshes at the same location, but then selecting a NavMesh becomes ambiguous.
- If you are loading a second NavMesh Surface additively and you have "dangling" links in the first scene, check that they do not connect to unwanted surfaces.

API Reference

NavMesh Surface

Properties

- *agentTypeID* – ID describing the agent type the NavMesh should be built for.

- *collectObjects* – defines how input geometry is collected from the scene, one of *UnityEngine.AI.CollectObjects:*
    - *All* – use all objects in the scene.
    - *Volume* – use all objects in the scene which touch the bounding volume (see *size* and *center*)
    - *Children* – use all objects which are children to the Game Object where the NavMesh Surface is attached to.
- *size* – dimensions of the build volume. The size is not affected by scaling.
- *center* – center of the build volume relative to the Transform center.
- *layerMask* – bitmask defining the layers on which the objects must be to be included in the baking.
- *useGeometry* – defined which geometry is used for baking, one of *UnityEngine.AI.NavMeshCollectGeometry*:
    - *RenderMeshes* – use geometry from render meshes and terrains
    - *PhysicsColliders* – use geometry from colliders and terrains.
- *defaultArea* – default area type for all input geometries, unless otherwise specified
- *ignoreNavMeshAgent* – true if Game Objects with a NavMeshAgent components should be ingnored as input
- *ignoreNavMeshObstacle* – true if Game Objects with a NavMeshAgent components should be ingnored as input
- *overrideTileSize* – true if tile size is set
- *tileSize* – tile size in voxels (the component desc has explanation how to choose tile size)
- *overrideVoxelSize* – true if the voxel size is set
- *voxelSize* – size of the voxel in world units (the component desc has explanation how to choose tile size)
- *buildHeightMesh* – Not implemented.
- *navMeshData* – reference to the NavMeshData the surface uses, or null if not set.
- *activeSurfaces* – list of all active NavMeshSurfaces

  Note: The above values affect how the bake results, thus, you must call BuildNavMesh() to make them count.

Public Functions

`void BuildNavMesh ()`

Builds a new NavMeshData based on the parameters set on NavMesh Surface. The data can be accessed via *navMeshData*.

NavMesh Modifier

Properties

- *overrideArea* – true if the modifier overrides area type
- *area* – new area type to apply
- *ignoreFromBuild* – true if the GameObject which contains the modifier and its' children should be not be used to NavMesh baking.
- *activeModifiers* – list of all active NavMeshModifiers

Public Functions

`bool AffectsAgentType(int agentTypeID)`

Returns true if the modifier applies to the specified agent type, otherwise false.

NavMesh Modifier Volume

Properties

- *size* – size of the bounding volume in local space units. Transform affects the size.
- *center* – center of the bounding volume in local space units. Transform affects the center.
- *area* – area type to apply for the NavMesh areas that are inside the bounding volume.

Public Functions

`bool AffectsAgentType(int agentTypeID)`

Returns true of the the modifier applies for the specified agent type.

NavMesh Link

Properties

- *agentTypeID* – the type of agent that can use the link.
- *startPoint* – start point of the link in local space units. Transform affects the location.
- *endPoint* – end point of the link in local space units. Transform affects the location.
- *width* – width of the link in world length units.
- *costModifier* – when the cost modifier value is non-negative the cost of moving over the NavMeshLink is equivalent to the cost modifier value times the Euclidean distance between NavMeshLink end points.
- *bidirectional* – if true the link can be traversed both ways, if false the link can be traversed only from start to end.
- *autoUpdate* – if true the link updates the endpoints to follow the transform of the GameObject every frame.
- *area* – area type of the link (used for pathfinding cost).

Public Functions

`void UpdateLink()`

Updates the link to match the associated transform. This is useful for updating a link explicitly after e.g. changing the transform position. It is not necessary if the *autoUpdate* property is enabled. However calling *UpdateLink* can have a much smaller performance impact if you rarely change the link transform.