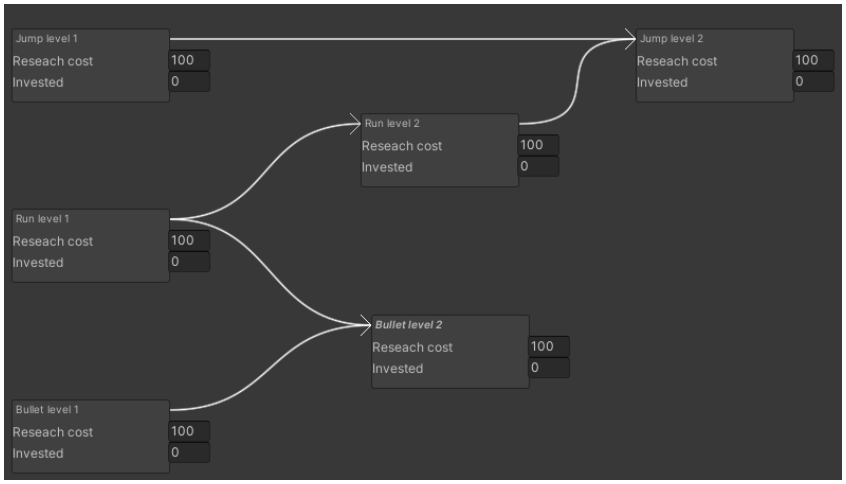
Manual: *Aoiti's Techtree / Skill tree Tool*



Aoiti's Techtree/Skilltree Creation Tool asset provides quick way to build a hierarchical network of nodes unidirectionally connected according to their prerequisites.

A technology or skill tree is the visual representation of the such network and the possible sequences of upgrades a player can take, usually through the act of research or experience points.

Using this tool, you will be able to,

1. create your own trees in the Techtree Editor Window by creating nodes and dragging connections,
2. assign UnityEvents to particular technologies/skills that will be invoked when research is complete,
3. access and update the upgrades in an individual techtree at runtime.

Structure

Aoiti's Techtree/Skilltree consists of three main classes that would concern a usual user. These are *Tech*, *Techtree* and *TechNode* classes.

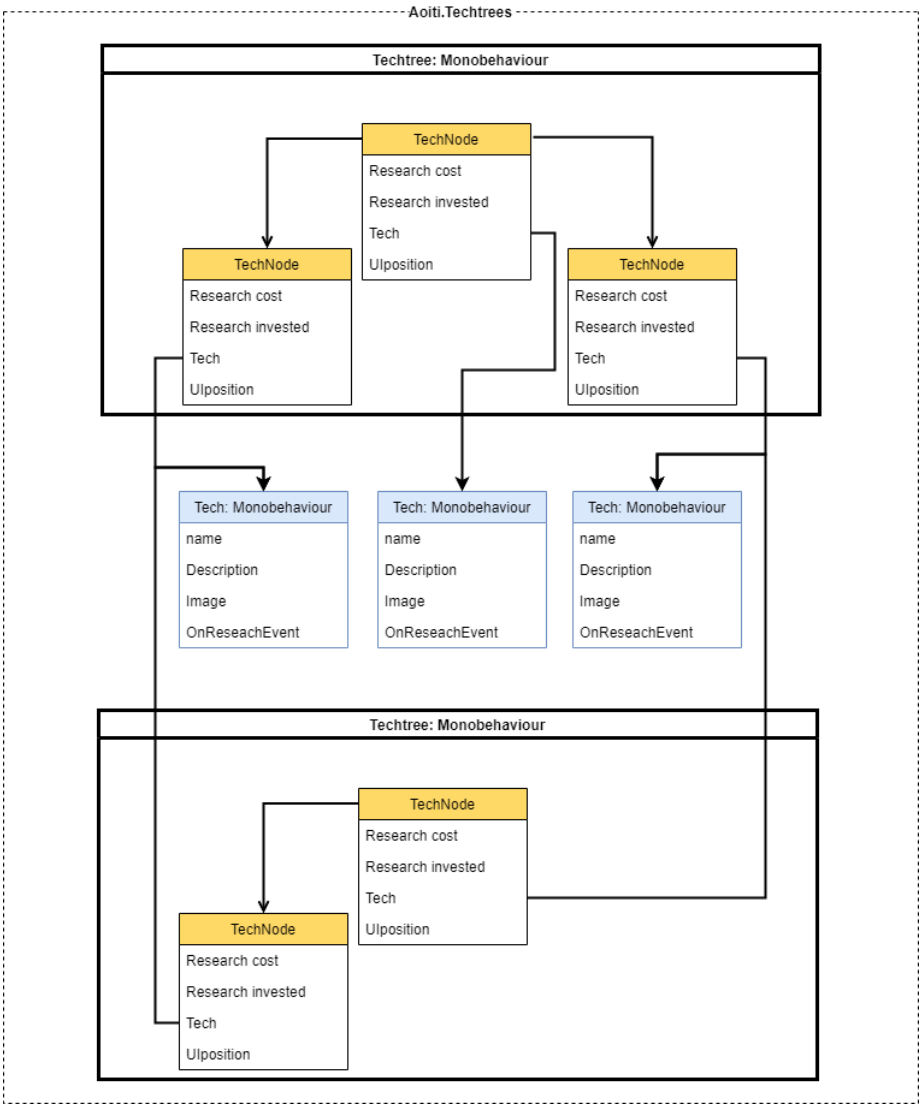
Tech is a monobehaviour that can hold, a name, description, image and a UnityEvent called *OnReseachComplete*. The *OnReseachComplete* can be subscribed by functions defined under the instances present in the scene. For more info on how UnityEvents work please visit Unity manual. Each *Tech* may be associated with more than one *Techtree* and *TechNode*.

TechNode (Node) represents the nodes in the Techtree/Skilltree and contains reference to a single associated *Tech*. Unlike *Techs*, Nodes exists only inside a *Techtree* object. They also contain how many points (as Research cost) are required to upgrade/complete and how many points are already invested (as Research invested). Each Node also contains a list of *Techs* required before starting to research this one. This is implicitly controlled, however can be ignored. Finally, it contains a Vector2, named

UIposition, to represent position of the UI element in the editor. *UIposition* can also be used to position the node in the game.

Techtree is a monobehaviour containing the list of nodes and necessary methods to manage and track the progress of the upgrades in the scene.

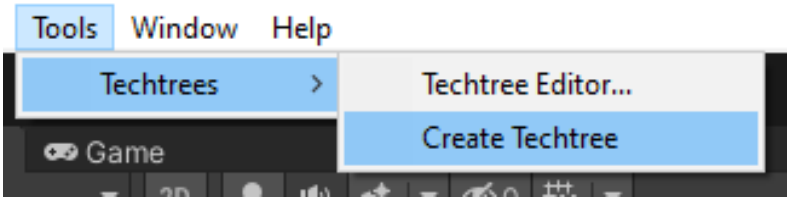
These classes are accessed using *Aoiti.Techtrees* namespace.



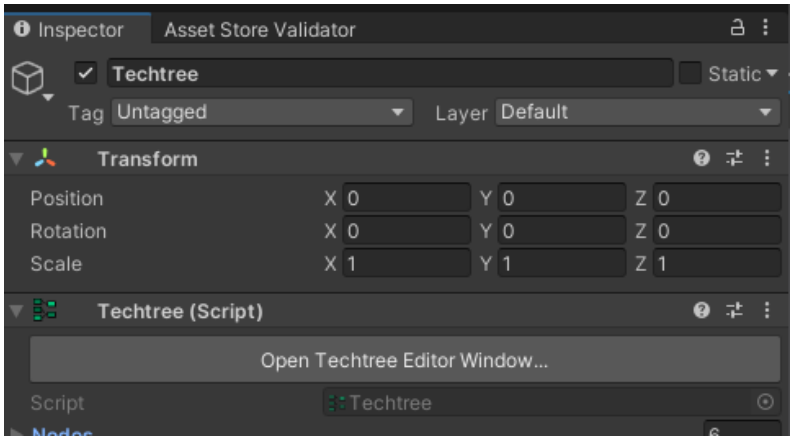
Creating a Techtree / Skilltree

This asset also contains necessary tools to create and edit *TechNodes* and *Techtrees*.

To start, use the “Tools\Techtrees\Create Techtree” menu item and create a gameobject with a *Techtree* component.

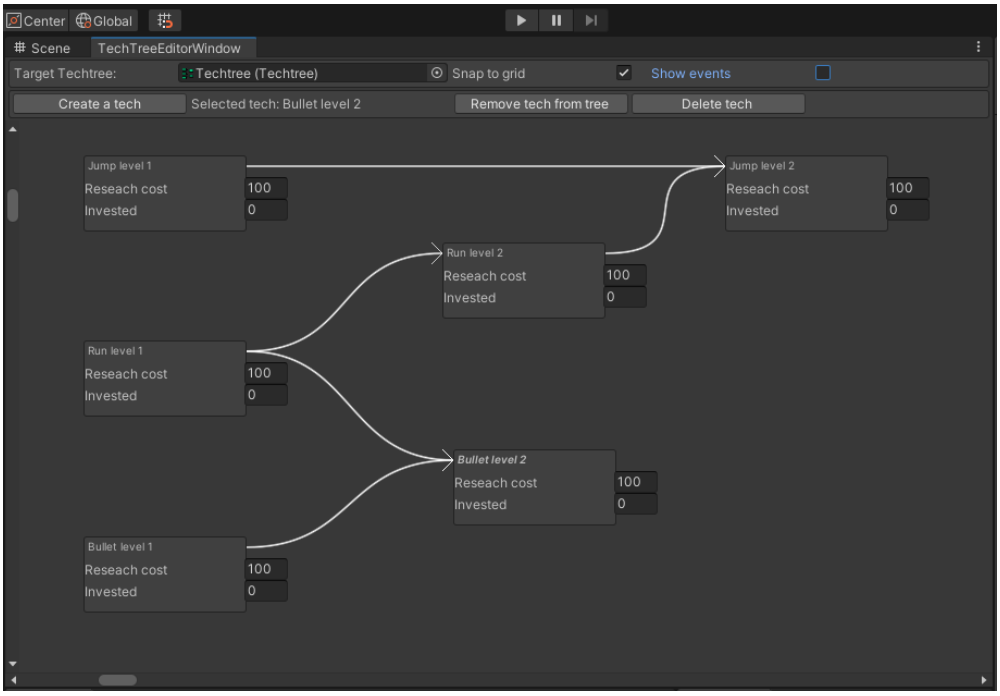


Select the newly created gameobject and start the *Techtree* Editor using “Open Techtree Editor Window…” button in the inspector. Alternatively, “Tools\Techtrees\Techtree Editor…” menu item also works.



Understanding Techtree Editor Window

The *Techtree* Editor Window opens as a new tab next to the Scene. In the editor window, you can visualize the nodes that represent each *Tech*.



The options in this window are listed as:

Target Techtree: It indicates the *Techtree* opened for editing. You can drag gameobject carrying the Techtree component to change the edited object.

Create a *Tech*: creates a gameobject with a *Tech* component, and loads onto the *Techtree* as a node.

TechNode (Node): Each *TechNode* is represented as a rounded box on the editor window, with the name of the *Tech* at the top.

Research cost: The total number of points required to complete the node.

Invested: The total number of points invested into the node.

Selected Tech: Indicate the currently selected *Tech*.

Remove Tech from tree: Removes the selected *Tech* from the tree.

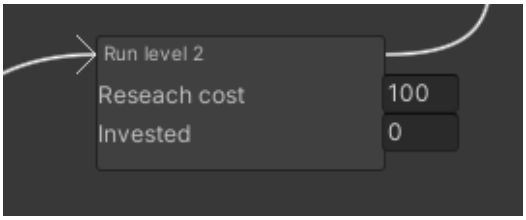
Delete Tech: Removes the selected *Tech* and deletes the associated gameobject. Please note that once deleted, cannot be undone.

Snap to grid: Snaps the nodes to a grid of size 10x10 during dragging.

Show events: Shows the *OnReseachComplete* UnityEvent under the node.

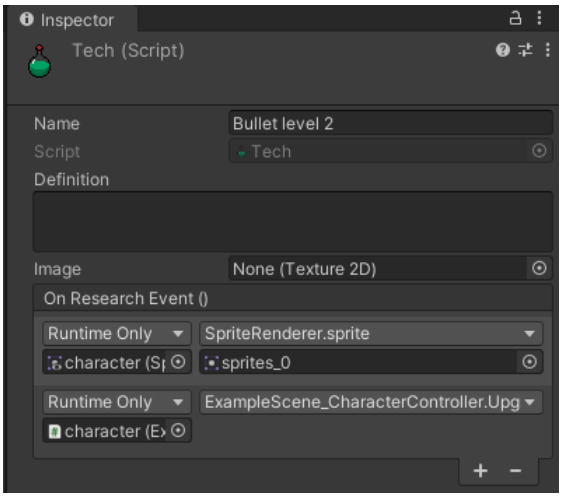
Creating *Techs* and nodes

Once a *Techtree* is selected, click on "Create a *Tech*" button to create a blank *Tech*. You can change the total research cost and initially invested research points inside the node. The research cost and invested points are independent from the Tech.



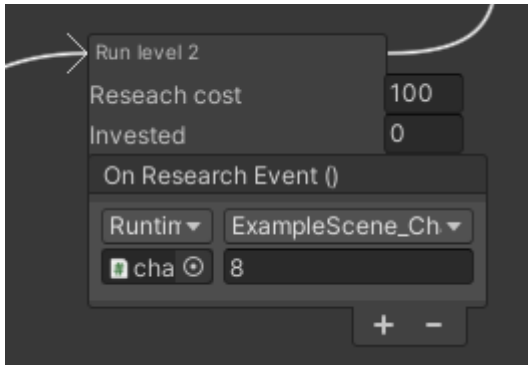
Editing *Techs*

Selecting a TechNode displays the associated *Tech* in the inspector. Changing the name automatically updates the name of the gameobject the Tech is connected to.



Assigning Functions to OnReseachComplete UnityEvent

You can also assign parameterless functions to *OnResearchComplete* UnityEvent through the editor or in the code using *AddListener* function. The OnResearchComplete is accessed through the inspector or, alternatively, by turning on "Show events" toggle in the editor. The latter option is designed to easily visualize all events in one window.




```
1. using Aoiti.Techtrees;
2.
3. technode.Tech.OnResearchComplete.AddListener(SomeFunction);
4.
5. technode.Tech.OnResearchComplete.AddListener(()=>return; //do
   stuff); //Lambda function example
```

The functions can also be assigned in the code as shown above. Please note that these functions need to be parameterless. As the case in UnityEvents, multiple functions may be assigned.

The *OnResearchComplete* events are invoked once the invested research exceeds the research cost. Alternatively, it can be called using *Invoke* function. Please note that using OnResearchComplete is not a requirement, and you can use the tool without this feature.

Loading already existing *Techs*

The *Techs* that are already in the scene can be loaded onto the *Techtree* by dragging and dropping the gameobject containing the *Tech* component into the editor window.

Removing or Deleting Techs

Techs may be removed temporarily or deleted permanently inside the editor window using “Remove Tech” and “Delete Tech” buttons, respectively. Please note that if a TechNode is associated with a prefab, it cannot be deleted, only removed.

Connecting/Disconnecting nodes

The hierarchy between the nodes are expressed as arrows originating from the requirement to the dependent. The dependencies have to be unidirectional. There can be multiple requirements for a dependent and a requirement can be required by multiple dependents.

To make dependency connections, click on the node that will be requirement using left mouse button. Then click on the dependent using the right mouse button. This will draw an arrow indicating the requirement. To remove an already existing connection, repeat the process.

Such connection will not be formed in following circumstances;

1. If the dependencies would form a cycle. The techtrees has to be unidirectional.
2. If the requirement is already required by another requirement of the dependent. In such case, the connection would be redundant, thus not allowed to save resources.

Similarly, in case a requirement in the downstream is added to a dependent that is connected to another requirement from the upstream of the techtree cascade, the former connection is removed for the more recent one.

Accessing *Techtree* by code

For a common techtree, the game needs to perform following at the runtime,

1. Iterate through TechNodes,
2. Check whether a TechNode is researched,
3. Check whether a TechNode has all its requirements met,
4. Invest research points to a Tech,
5. Detect if a Tech is research.

To iterate through the nodes, use the *IterateTechNodes* function as follows;

```
1. using Aoiti.Techtrees;
2.
3. foreach (TechNode technode in techtree.IterateTechNodes())
4. {
5.     //do stuff
6. }
```

To check whether a *Tech* is researched, use *IsResearched* function as follows;

```
1. using Aoiti.Techtrees;
2.
3. bool researched = techtree.IsResearched(SomeTech);
4.
5. //OR
6.
7. bool researched = techtree.IsResearched
```

To check whether all requirements of a Tech is met, use *RequirementsMet* function as follows;

```
1. using Aoiti.Techtrees;
2.
3. bool reqs_done = techtree.RequirementsMet(SomeTech);
4.
5. //OR
6.
7. bool reqs_done = techtree.RequirementsMet(SomeTechNode);
```

To invest points to a technode, use *Research* function as follows;

```
1. using Aoiti.Techtrees;
2.
3. bool researchComplete=techtree.Research(SomeTech,SomePoints);
4.
5. //OR
6.
7. bool researchComplete=techtree.Research(SomeTechNode,SomePoints);
```

Research function returns true only if the *Tech* is already researched or just completed researching. This function implicitly confirms that all requirements were already met, before investing points, otherwise does not do anything. To explicitly invest points regardless the requirements, use *ignoreRequirements=true* argument as followed;

```
1. using Aoiti.Techtrees;
2.
3. bool researchComplete= techtree.Research(SomeTech,SomePoints,
   ignoreRequirements=true);
4.
```

```
5. //OR
6.
7. bool researchComplete= techtree.Research(SomeTechNode,SomePoints,
   ignoreRequirements=true);
```

Detection of Whether a Tech is completed researching, may be done in three ways,

1. Iterate through the Techs inside each turn, *Update*, *FixedUpdate* or another periodically invoked function and check whether a *Tech* is researched using *IsReseached* function.
2. Assign functions to *OnResearchComplete* event that will perform a desired effect upon completion.
3. Check the output of the *Research* function when investing points.

Modifying *Techs* at script-level

For reasons, programmers may want to add more features to the *Tech* or *TechNodes*. In that case, the *TechNode* class may be found at the beginning of "Techtree.cs". *Tech* class is contained in its own C# script file, "Tech.cs". You can recognize them by these icons;

"Techtree.cs" icon:	"Tech.cs" icon: