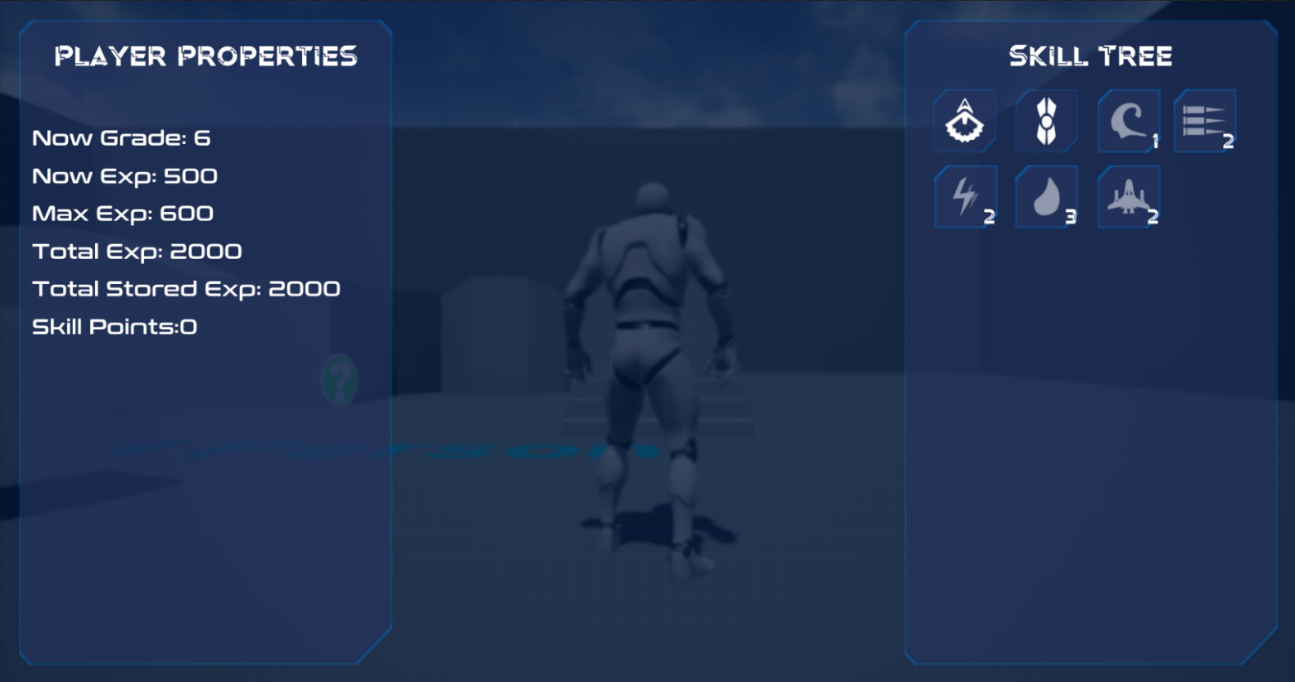
Skill Tree System

Owen Chen - c019220j

**Contents**

----------------------------------------------------------------------------------------------------------------------

[Description: 1](#_Toc90208888)

[Researches: 3](#_Toc90208889)

[Requirements: 5](#_Toc90208890)

[System Design: 6](#_Toc90208891)

[Loading Process: 6](#_Toc90208892)

[Skill Tree Show / Hide: 7](#_Toc90208893)

[Node Activation: 7](#_Toc90208894)

[Experience: 7](#_Toc90208895)

[User Guide: 8](#_Toc90208896)

[Step 1: 8](#_Toc90208897)

# Description:

Some skills can only be learned by pre-learning specific skills, talents and by reaching the appropriate level and attribute requirements.

For games with a skill point (talent point) mechanism, depending on the game system, skill points can be obtained through leveling up and game quests, and there are also props and NPCs for adding/cleansing skills.

A tree diagram of the links between skills is known as a "skill tree".

Examples:

* Mages who specialize in ice ability skills must learn basic ice attack spells (such as ice arrows) before they can learn advanced ice attack spells such as ice blasts and blizzards.
* Each profession in World of Warcraft has three profession talents. Some profession skills are common to that profession, but some skills must be acquired by taking a specific talent and by investing a specified number of points in that talent.
* Some games have a mechanic where more than two different career skills can be tapped, offering the player the freedom to figure out and assign the desired career skill and fighting style. An example of this is the Titan Quest series.

In my project I will be creating a Skill Tree System that provides a trigger API that can bind various skills, effects and attributes for designers to use.

As the player progresses, the player will be able to unlock nodes in the skill tree so skills, effects and attributes are available to the player.

Most RPGs have an experience system that can be paired with a skill tree to enrich the player's game. There are many examples of successful games in the history of game design that have used a Skill Tree System.

E.g.: World of Warcraft, Diablo 2, and League of Legend.



# Researches:

What is a Skill Tree? Here is a common explanation:

In strategy games, a technology, tech, or research tree is a hierarchical visual representation of the possible sequences of upgrades a player can take (most often through the act of research). Because these trees are technically directed and acyclic, they can more accurately be described as a technology directed acyclic graph. The diagram is tree-shaped in the sense that it branches between each 'level', allowing the player to choose one sequence or another. Each level is called a tier and is often used to describe the technological strength of a player. Typically, at the beginning of a session of a strategy game, a player will start at tier 1, and will only have a few options for technologies to research. Each technology that a player researches will open up one or more new options, but may or may not, depending on the computer game, close off the paths to other options. The tech tree is the representation of all possible paths of research a player can take, up to the culmination of said sequence. - [Technology Tree Wikipedia](https://en.wikipedia.org/wiki/Technology_tree)

The skill tree system I have been researching on is Path of Exile.

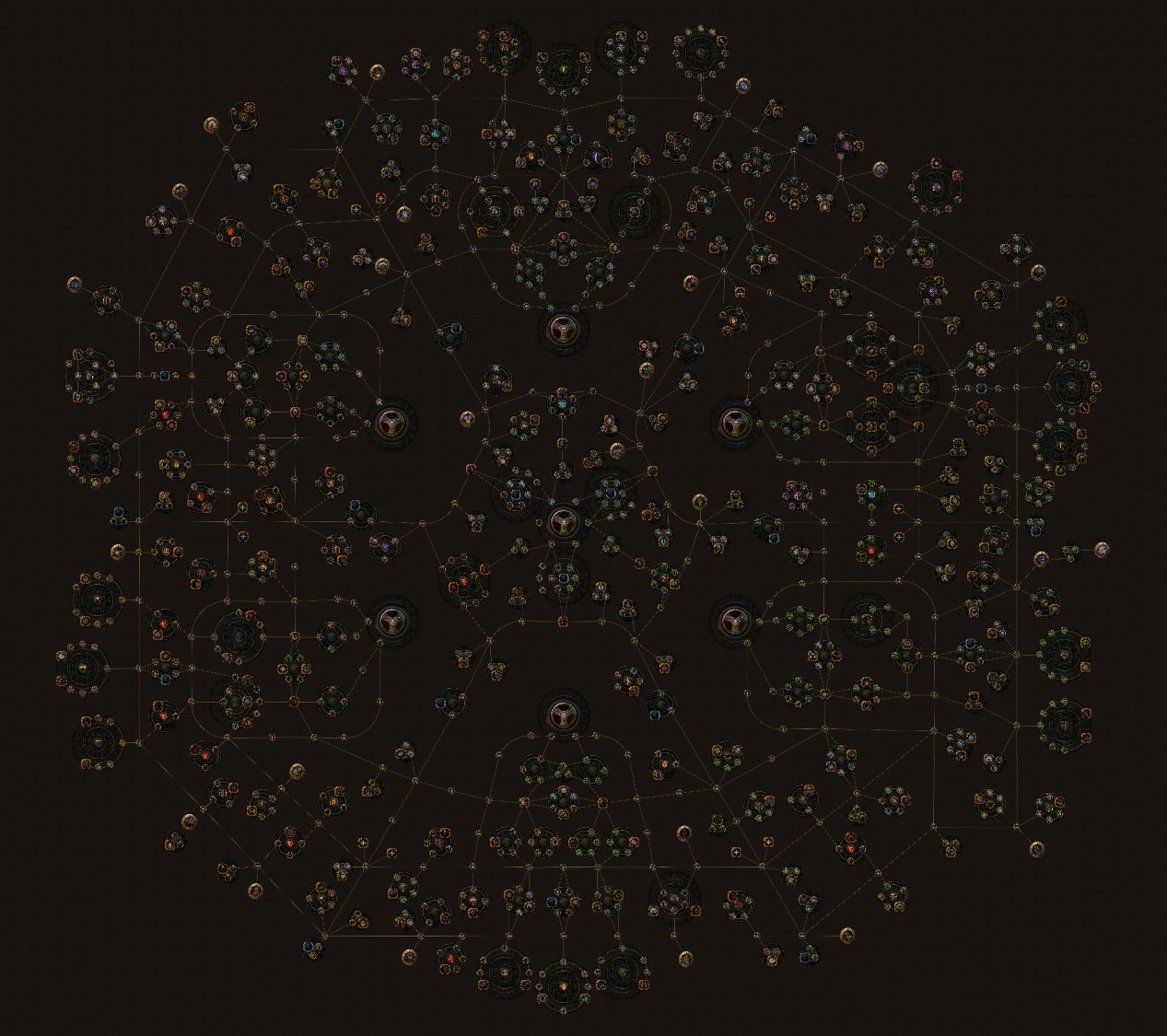
*-* [*Passive Skill Tree - Path of Exile*](https://www.pathofexile.com/passive-skill-tree)

In Journey of the ***Titan Quest***and ***Grim Dawn***, the player has to choose a line of professions' talents soon after character creation and then they can't change, after a while the game will unlock a second talent and you can choose another one from the remaining professions, so they are stricter and although there are plenty of trees, you can only add 2 choices, unlike ***Diablo 2*** or ***Dungeon Siege 2*** which are so free.

For traditional RPGs with professions (Classes), it's destined to be a big pile of trees that are many and shallow. But from an ARPG history point of view, players soon got tired of such games with pre-defined professions. The Japanese RPGs were shrewd enough to introduce the concept of 'transferability', whereby the basic class could be transferred into multiple advanced classes, meaning that these games had not only a skill tree but also a 'career tree', thus increasing the playability of the game. Another great design in Japanese RPGs is the hybrid profession, where the player can choose another profession as a secondary profession, thus increasing the complexity of the skill tree to improve the playability of the game.

There is also the more barbaric way of simply piling up the number of professions, a dozen professions, hundreds of professions...... are really not worth mentioning just to leave it out.

The European and American RPGs have gone the other way, the design of free professions (Class-Free). Rather than giving you multiple shallow trees for you to pick and choose from there for combinations, it is better to go one way to the end, such as giving you a this.



Path of Exile

In Path of Exile, its skill tree doesn’t contain any active skills, and it can only put in one token per node, and each of his nodes requires the same number of tokens, but you’ll find a lot of three of four that make up a cluster of skill points, and one of them is always the better one, so that in effect you are aiming for the one optimal one, and the others are all passing points.

So, summaries what observed from these examples, my system is going to have:

1. Tokens / points to unlock nodes.
2. Trigger that could use to unlocks Active Skills for the player.
3. Nodes that can repeatedly invested in tokens / points.
4. Good foundation of readability and aesthetics.
5. An Attributes System of the character could be pair with Skill Tree System.
6. Multiplayer implementation.

# Requirements:

Skill Tree that works through components attached to the player controller/pawn with hooks for adding points and a base hierarchy for the skills themselves as template for end user expansion.

Adding dynamic UI and tools for designers to affect attributes on the attached actor with drag-and-drop.

# System Design:

## Loading Process:

The Skill Tree Actor could be inherited or used directly as a Child-Actor within the Player Character (recommended).

The Skill Tree Actor contains APIs for the Experience modification, initialization of the Skill Tree Widget and references of the Skill Tree Component.

The Skill Tree Component holds runtime variable data of the Experience, the Skill Points (Tokens) and the reference of the Skill Tree Widget.

To load up the whole system, you need to fire the DoLoad function within the Skill Tree Actor.   
Which construct the Skill Tree Widget and add it to the player’s viewport. Then, bind the SyncLocalDataAndStoredData function to OnTick delegate to syncs the data in the storage with runtime data.

When constructing the Skill Tree Widget, all Skill Tree Nodes on the canvas will be saved to C++-accessible temporary variables. The activation data for the nodes is then loaded from USaveGameData (Storage) and the Skill Tree Widget will be updated. Updated nodes will be saved to storage in case there are updates.

Finally, the Skill Tree Actor fires DoStart function within the Skill Tree Component. It holds the references of the SkillTreeInstance for accessing the storage, and SkillTreeUI used to Show / Hide the widget.

## Skill Tree Show / Hide:

The player calls the SkillTreeComponent in the Child-Actor (Skill Tree Actor) to activate the OpenSkillTree function.

The default show and hide are controlled by a fade animation that changes the render opacity of the entire widget canvas.

## Node Activation:

The player clicks the Activate button, the system attempts to activate the skill and returns the result, the system analyses the result to determine if it is activated, and if so, broadcasts the Node Activation delegate.

Designers can bind events to the Node Activation delegate and use them for play upgrade effects, play upgrade audio, etc.

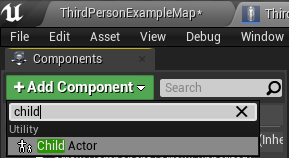
## Experience:

The player calls experience modification functions and saves runtime data to storage.

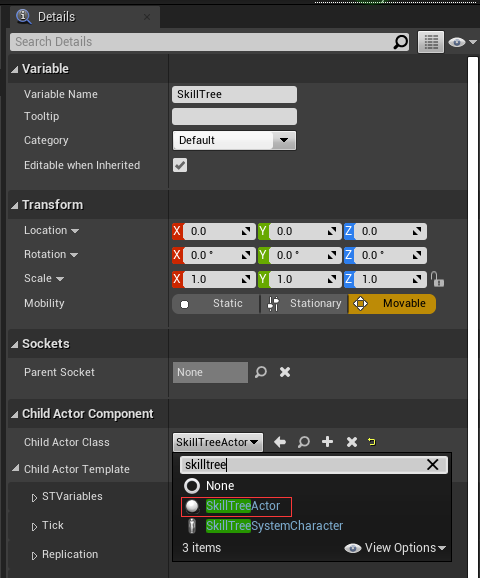
# User Guide:

## Prepare the system:

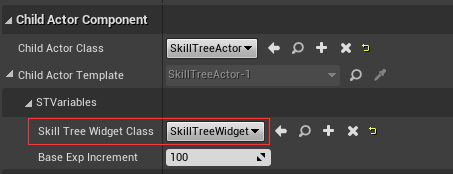
### *Step 1:*

Create a child actor in your player character (recommended).

### *Step 2:*

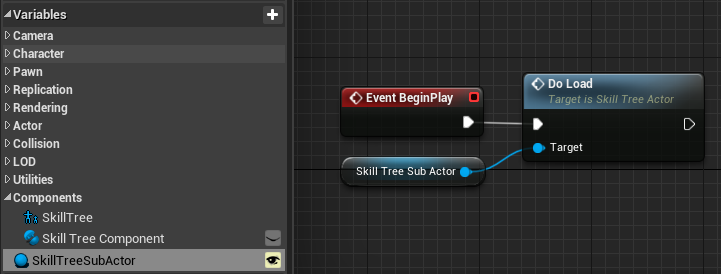
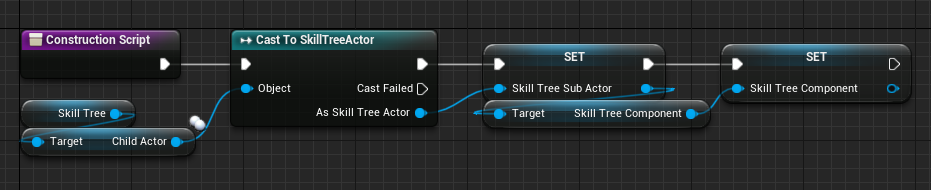
In detail panel of child actor, find Child Actor Component page, and select Skill Tree Actor as child class.

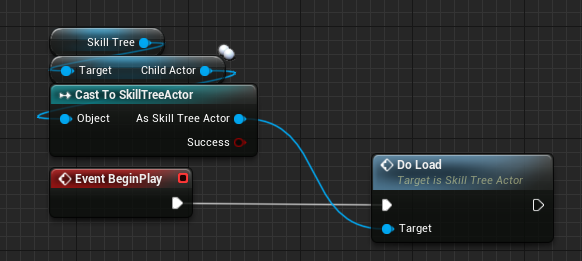
### *Step 3:*

Reference the Skill Tree Widget class.

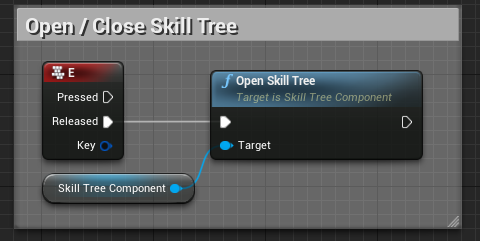
And change the Base Exp Increment value if you want.

### *Step 4:*



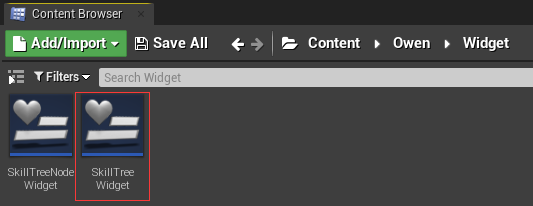
You could save the reference of Child Actor’s class to variables when your player character is constructing. Or access the DoLoad function directly. 

This could be updated to a delegate that broadcasts DoLoad, and the system receive and run the DoLoad function behind the scenes.

And the show / hide of the Skill Tree is controlled by the function within the Skill Tree Component.

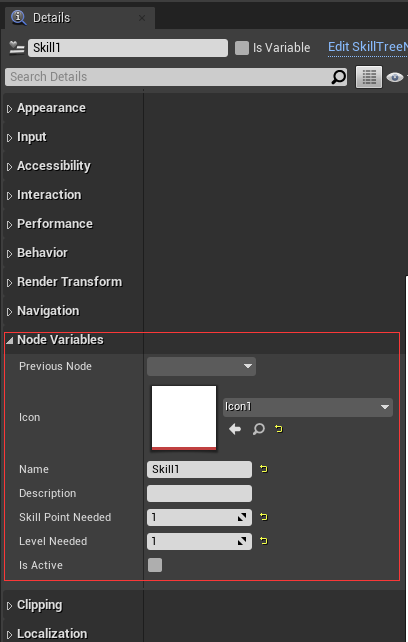
## Adding new Skill:

### *Step 1:*



Find the Skill Tree Widget class, open its designer page.

### *Step 2:*

On the hierachy, there is a Tree Nodes canvas which holds all the Skill Nodes.

Simply duplicate one of them, and on the details page find Node Variables page to set-up the skill.

**NOTE: NAME CANNOT BE REPEATED.**

**Improvements could be done:**

Creating a data object that stores the data required for a skill and dynamically loading the data object into the Skill Tree Node will make it easier for designers to locate and edit the data.