Achievement System

User Guide

Overview

Achievement System is a tool which allows developers to easily create and manage in-game achievements.

Features

- Progress and Goal achievements.
- Intuitive editor window for creating and monitoring achievement values.
- Ability to hide spoiler achievements from the player.
- In-game menu to show the players progress.
- Automatic achievement saving between game loads.
- Customisable progress display frequency.

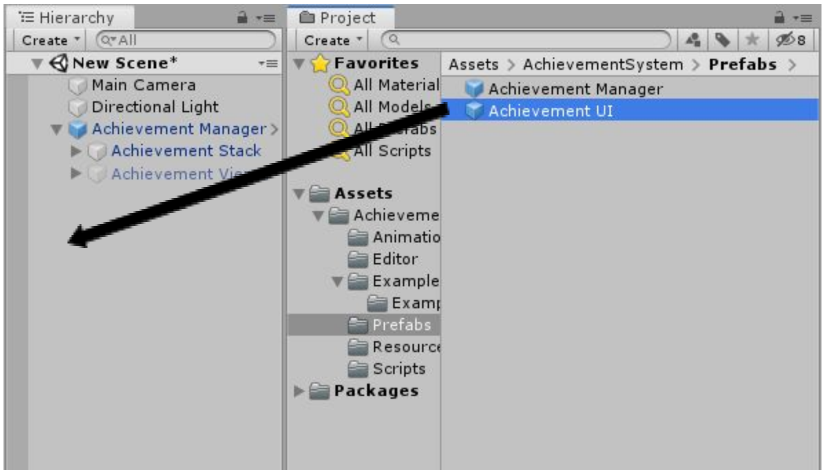
Getting Started

Achievement Manager Prefab

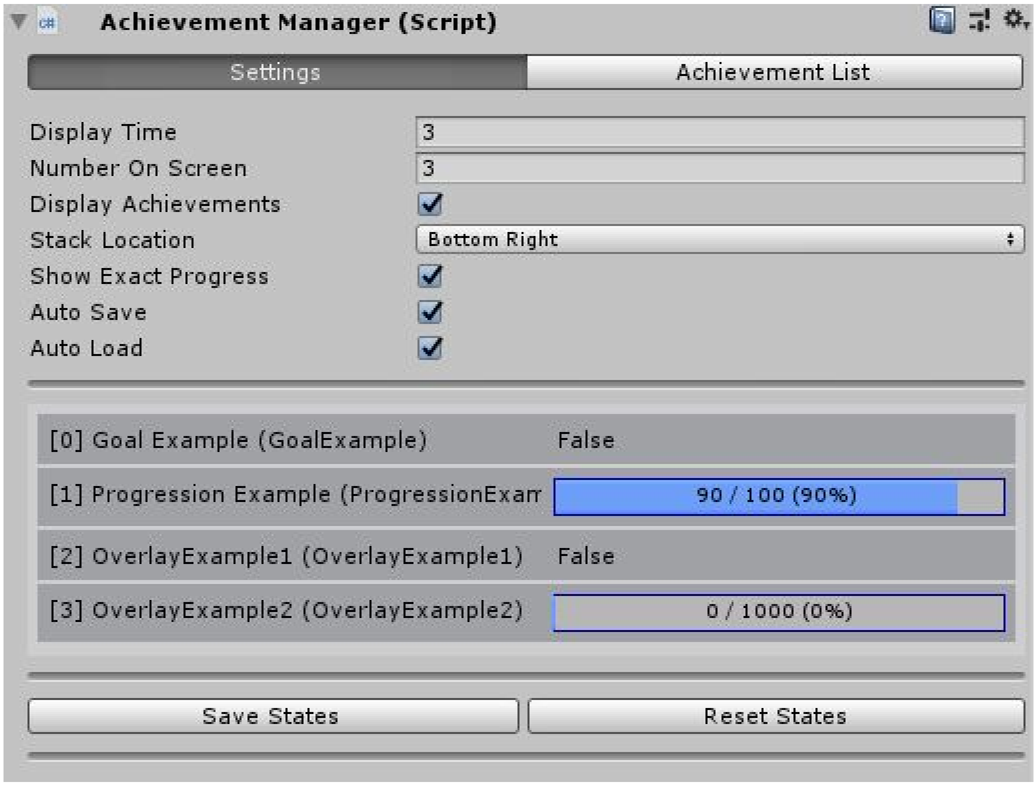
To use the achievement system the starting scene must include an Achievement Manager object.

This can be found at : **/Assets/AchievementSystem/Prefabs/**

The Achievement Manager is the object responsible for all developer interaction with the system. Note - This item only needs to be included once as it will remain persistent through scene loads.



Achievement Manager Settings



Display Time	3
Number On Screen	3
Display Achievements	☑
Stack Location	Bottom Right
Show Exact Progress	☑
Auto Save	☑
Auto Load	☑

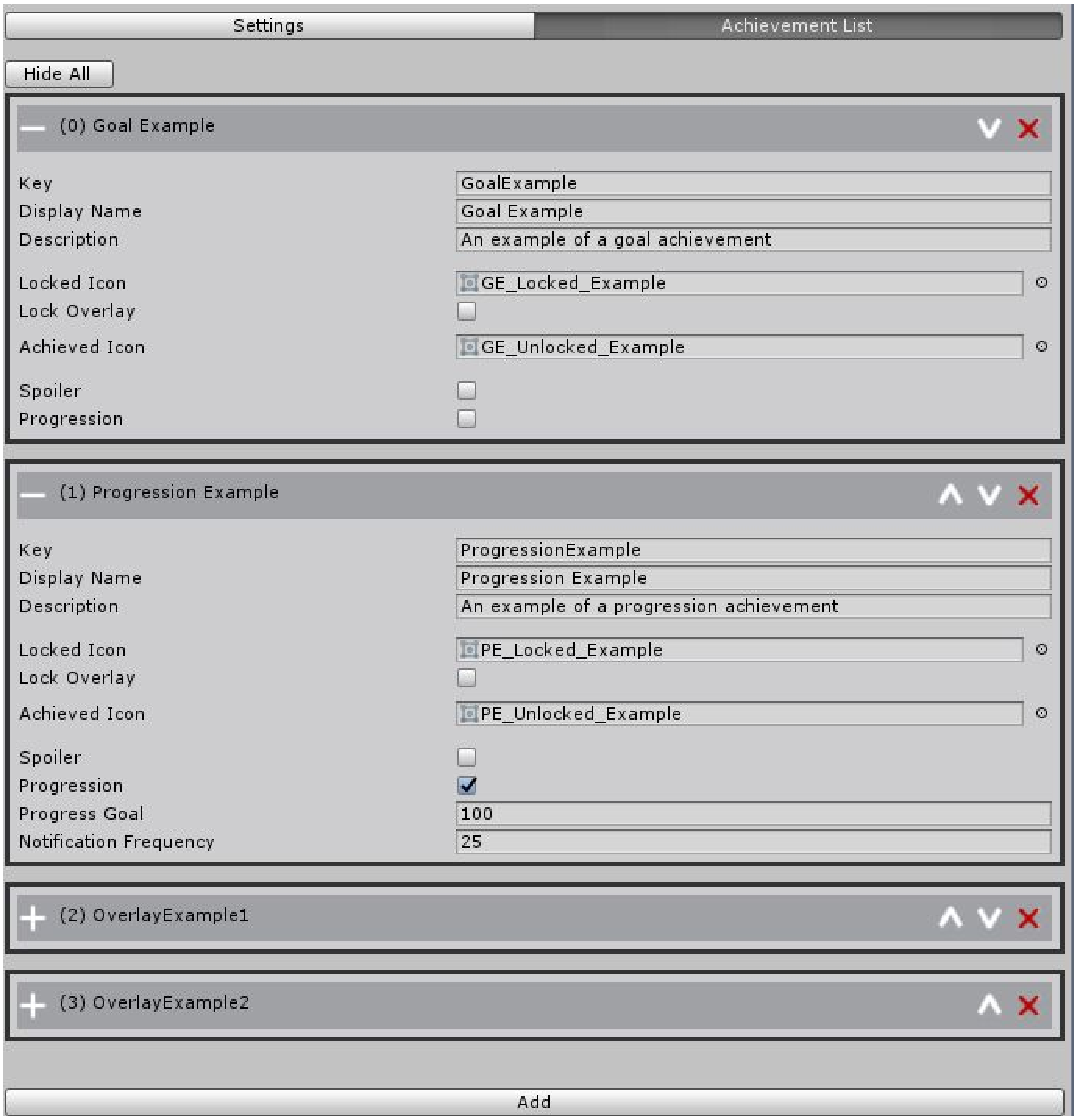
[0] Goal Example (GoalExample)	False
[1] Progression Example (ProgressionExam	90 / 100 (90%)
[2] OverlayExample1 (OverlayExample1)	False
[3] OverlayExample2 (OverlayExample2)	0 / 1000 (0%)

Within the first tab, there is a list of settings which can be used to adjust the behaviour of the manager.

Setting	Description
Display Time	The number of seconds an achievement will display on the screen once unlocked or progress is made.
Screen Count	The total number of achievements which can be on the screen at any one time.
Display Achievements	If **true,** achievement unlocks / progress update notifications will be displayed on the players screen.
Stack Location	The place on the screen where notifications will be displayed.
Show Exact Progress	If **true**, progress notifications will display their exact progress if false it will show the closest bracket. Example - Notification frequency is set to 25 and the current progress is 26. true = Display 26 false = Display 25
Auto Save	If **true,** the state of all achievements will be saved without any user

	input. If **false,** achievement states must be saved manually by calling SaveAchievementState(). Set to false improve performance.
Auto Load	If **true,** the state of all achievements will be automatically loaded when the game starts. If **false,** achievement states must be loaded manually by calling LoadAchievementState().

Achievement Manager List



(0) Goal Example

Key: GoalExample
Display Name: Goal Example
Description: An example of a goal achievement
Locked Icon: GE_Locked_Example
Lock Overlay: □
Achieved Icon: GE_Unlocked_Example
Spoiler: □
Progression: □

(1) Progression Example

Key: ProgressionExample
Display Name: Progression Example
Description: An example of a progression achievement
Locked Icon: PE_Locked_Example
Lock Overlay: □
Achieved Icon: PE_Unlocked_Example
Spoiler: □
Progression: ☑
Progress Goal: 100
Notification Frequency: 25

(2) OverlayExample1

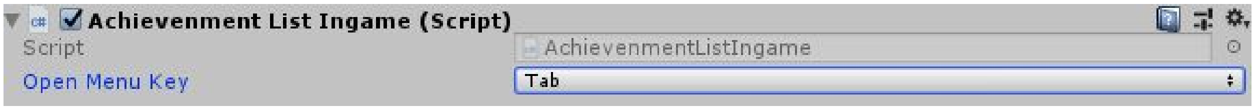
(3) OverlayExample2

The second tab is where the list of achievements that the player can earn are stored. To create a new achievement the add button can be clicked. The navigation button and X can be used on each element to rearrange / remove elements from the list. Additionally, the - and + buttons on the left of each element will show and hide the achievement to increase visibility.

Setting	Description
Key	A unique key used to identify an achievement in code.
Display Name	The name of the achievement which the player will see in-game.
Description	A short description which tells the player how to earn the achievement.
Lock and Unlocked Icons	The icon which will be displayed when the achievement is locked and unlocked. If **Lock overlay** is set to true the lock icon will be overlayed on top of the Unlocked version. See in example scene.
Spoiler	Treat the current achievement as a spoiler for the game. Hidden from player until unlocked.
Progression	If true this achievement will count to a certain amount before unlocking. E.g. race a total of 500 km, collect 10 coins or reach a high score of 25.
Progress Goal	The goal which must be reached for the achievement to unlock. Used only for progression based achievements.
Notification Frequency	The rate that progress updates will be displayed on the screen e.g. Progress goal = 100 and Notification Frequency = 25. In this example, the progress will be displayed at 25,50,75 and 100%.

Ingame Achievement viewer

The settings for the in-game viewer can be found on the Achievement Manager prefab. You can set the key that will be used to open the in-game menu here.

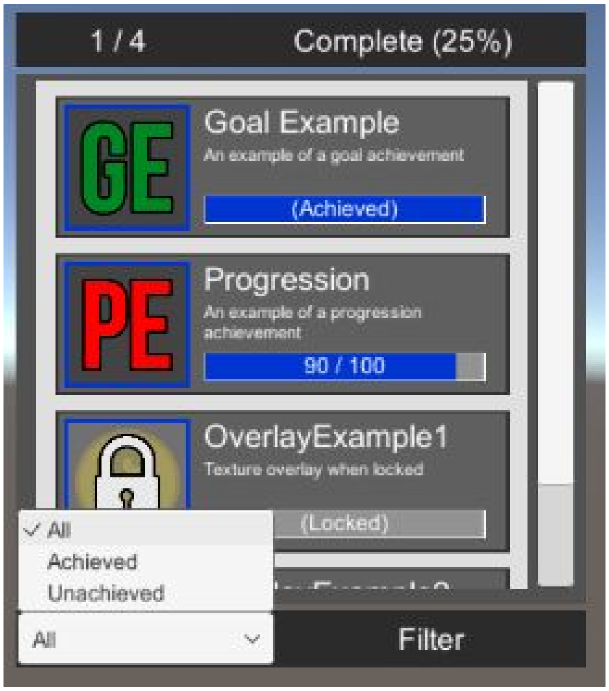


☑ Achievenment List Ingame (Script)

Script: AchievenmentListIngame

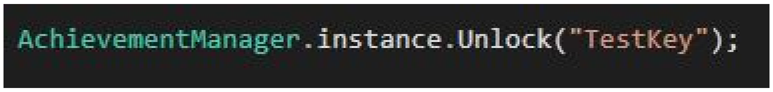
Open Menu Key: Tab

The in-game viewer will show the player a list of achievements, their progress and allow them to filter them based on if they are achieved or not.



Unlocking Achievements through Code

To unlock/add progress to achievements there a several functions which can be called. All unlocking/progress functions have two options of access including by **Index** or **Key**. All functions are called by accessing the achievement manager singleton as seen below.



```
AchievementManager.instance.Unlock("TestKey");
```

(Example to unlock and achievement with the key "TestKey")

A detailed description of all public functions can be found below.

Function Reference

The game manager includes 9 (+3 overloads) public functions which can be called to interact with the system.

Miscellaneous

```
public bool AchievementExists(string Key){}
public bool AchievementExists(int Index){}
```

Returns true if an achievement is found in the list.

```
public int GetAchievedCount(){}
```

Returns the total number of achievements which have been unlocked.

```
public float GetAchievedPercentage(){}
```

Returns the current percentage of unlocked achievements.

Unlock and Progress

```
public void Unlock(string Key){}
public void Unlock(int Index){}
```

Fully unlocks a progression or goal achievement.

```
public void SetAchievementProgress(string Key, float Progress){}
public void SetAchievementProgress(int Index,  float Progress){}
```

Set the progress of an achievement to a specific value.

```
public void AddAchievementProgress(string Key, float Progress){}
public void AddAchievementProgress(int Index,  float Progress){}
```

Adds the input amount of progress to an achievement. Clamps achievement progress to its max value.

Saving and loading

```
public void SaveAchievementState(){}
```

Saves progress and achieved states to player prefs. Used to allow reload of data between game loads.
This function is automatically called if the **Auto Save** setting is set to true.

```
public void LoadAchievementState(){}
```

Loads all progress and achievement states from player prefs.
This function is automatically called if the **Auto Load** setting is set to true.

```
public void ResetAchievementState(){}
```

Clears all saved progress and achieved states.