What is Narrate?

Narrate is the quick and flexible solution for adding a narrator to your game. Putting a witty line of dialogue in your scenes should be easy, and with Narrate it is. For many games you'll find the prebuilt components are more than enough to implement your storyteller without writing a line of code. But if you find yourself in need of functionality unique to your game, never fear - the simple, modular system is easily extensible.

Need subtitles? Don't worry. We've got you covered, and the display is easily customizable, too.

Features:

- A self-contained Narration system that can be added to just about any game without interfering with the systems already in place.
- Prefabs that allow you to get your system up in minutes.
- Several pre-built triggers that will allow you to get a non-branching dialogue or a narrator like those in *Thomas Was Alone* and *The Stanley Parable* up and running without a line of code.
- Self-scaling subtitle display that can easily be customized to match the rest of a game's UI.

Overview

The Narrate system is built from 5 primary components:

1. Narrations

   Narrations hold the audio clip and its accompanying subtitle(s).

2. The Managers

   The Narration Manager is the system's beneficent overlord. It manages the actual playing of the audio clips, makes sure that narrations don't interrupt each other unless they have permission, and orders its minion, the Subtitle Manager, to display the subtitle bar if subtitles are enabled.

3. Triggers

   Triggers are a more feature-rich wrapper class for Narrations. They detect when a Narration needs to be played and sends that Narration to the Narration Manager. Or, they may call on a NarrationList to play instead.

4. NarrationLists

   A NarrationList is a pool of Narrations which can be played from. NarrationLists allow randomization of Narrations, looping, and a few other neat little tricks.

5. The Subtitle Display

   The display is the customizable UI element where any subtitles are displayed when subtitles are enabled.

Quick-Start Guide:

- Add the one of the NarrationSystem prefabs from the Narrate->Prefabs folder to the scene/heirarchy.
- Make an empty GameObject. Attach a ProximityNarrationTrigger script to it.
- Setting up the ProximityNarrationTrigger:
  - Expand the Narrations foldout and the Phrases foldout. Click the "+" button to add a new Phrase.
  - Drag audio in if you have any, and/or type in a line of text.
  - Set the phrase's duration to however long the Phrase should be played for.
  - Expand the ProximitySettings foldout.
  - Set Proximity to 5;
  - Set Target to your player's gameobject's transform.
- Now, when your player's transform is within 5 units of the ProximityNarrationTrigger's gameobject, the Narration will play.

Congratulations! You've added your first Narration to your game, but you've only scratched the surface. Read this manual and check out the demo scenes to see what else you can accomplish with Narrate.

The Demos

Two simple demo scenes, one 2D and one 3D, are included. Note, however, that the Narrate system in no way depends upon whether it is being used in 2D or 3D environments.

The demos can be found in the *Narrate->Demo* folder or online at www.jackofhacks.com/unityAssets/unityAssets.html

Phrases

A Phrase is the subatomic particle of the Narrate system - they're a data type that you'll likely never work with directly, but still use frequently. They contain an AudioClip called *Audio*, a string called *Text*, and a float called *Duration*. *Audio* is what the sound will be played when this Phrase is played, *Text* is the text/subtitle that will display, and *Duration* is how long the Audio and/or Text will be allowed to play or be displayed for.

It is not necessary for both the *Audio* and *Text* fields to be filled - you may pick one or the other. If *Duration* is left set to zero (and SmartSubs are not enabled) the Phrase will be played for the *DefaultPhraseDuration,* which is set by the *NarrationManager.*

A lone Phrase can represent one speech/sentence, or several can be combined to represent one longer sentence/speech. For example, you can simply give each Phrase in a Narration its own AudioClip and Subtitle, and thus each is an individual bit of speech. On the other hand, you can give your Narration a single, long AudioClip and then use the Phrases as segmented subtitles so you don't overwhelm the SubtitleDisplay and its readers.

Narrations

Narrations are the atom of the Narrate system - it is the smallest, playable data type. Anything that is played by the Narrate system is stored in, or represented by, a Narration.

Components:

1. Phrases -

   Each Narration contains an array of Phrases. If *Single Audio Clip* is enabled, it will appear under the name "Subtitles"

2. A BusyBehavior

   A BusyBehavior notes what should happen to this Narration if there is

already a Narration being played. We can never play more than one Narration at once - that wo. So instead, maybe we should just wait until whatever is already being played is done before playing this one. Or maybe we want to interrupt because this one is very important. Or maybe we just won't play this one at all. BusyBehaviors let you decide what should happen. They include:

- *Queue* - The Narration will be put at the end of the line of Narrations waiting to be played. If there is no room in line, nothing will happen. It will not be added.
- *PrioritizeInQueue* -The Narration will be put at the front of the line of Narrations waiting to be played so that it can be played next.
- *Interrupt* - Whatever is currently being played will be stopped and discarded, unfinished. This Narration will be played instead.
- *DoNothing* - The Narration won't be played or queued.

3. Loop Audio Per Phrase/Subtitle (Option)

   If enabled, this will cause the provided AudioClip to play on loop for each Phrase for the Phrase's duration. For example, if you have a 0.5 second retro "blip" sound that should play whenever a certain character talks, and their Phrase last 6 seconds, the sound would be looped for the whole 6 seconds.

   This functions the same regardless of whether or not the Narration has *Single Audio Clip* or *Smart Subs* enabled.

4. Single Audio Clip (Option)

   If enabled, this prevents individual Phrases from receiving AudioClips. Instead, a new AudioClip field will appear for the single AudioClip that will be used for the entire Narration.

   If *Loop Audio Per Phrase* is not enabled, this single AudioClip will be played once and the Phrases will act as its subtitles. This is great for providing more-manageable subtitles for readers when using long audio files.

   If *Loop Audio Per Phrase* is enabled, this single AudioClip will be looped for each Phrases' duration. Thus, it might play several times, once, or never quite finish - all control rests with the Phrases.

5. Smart Subs (Option)

   If enabled, this will decide how long each Phrase's duration should be automatically, making the *Duration* field for the Phrases disappear.

   If *Loop Audio Per Phrase* is enabled, if a Phrase has no AudioClip, or if the Narration is set to *Single Audio Clip* but has no main AudioClip, the

Phrase's duration will default to the *DefaultPhraseDuration* which is assigned in the NarrationManager.

If the Narration is set to *Single Audio Clip* and has been given its single AudioClip, each Phrase will get an equal fraction of that AudioClips length.

Coding Tip:
A Narration is played by calling NarrationManager's `PlayNarration(Narration toPlay)` function. This is further covered in the section on the Narration Manager.

The Narration Manager

The NarrationManager is DJ-overlord of the Narration system. It controls the speakers (AudioSource) over which all Narrations are played.

The NarrationManager only allows one Narration to be played at a time. The manager maintains of queue of Narrations waiting to be played as well. This queue's length is finite to prevent spamming, and can be set via the NarrationManagers *Queue Length* field. The minimum length is one, to hold the one Narration it's playing.

It's generally a good idea to design your game so that the queue rarely has more than 1 or 2 Narrations in it, simply because back-to-back clips of unrelated dialogue usually don't sound very good. It's better to just drop that witty line the narrator wanted to say 5 minutes ago when the player collected their 100th flag, because the player's long forgotten about it.

If a new Narration wants to be played but the NarrationManger is already busily playing something, 3 things can happen depending on the new Narration's BusyBehaviour and how full the queue is:

1. Nothing - the new Narration will be ignored
2. Queued - the new Narration will be added to the queue, or list, of Narrations waiting to be played, either at the front or back. Look at the Narration Section for more info on "Queuing" BusyBehaviours.
3. Interrupt - the Narration already being played is stopped and discarded. The new Narration is immediately played.

The NarrationManager's *Default Phrase Duration* specifies how long a Phrase received with a duration of 0 will be played, as Phrases cannot be displayed for non-positive durations.

The NarrationManager also has a foldout called "Narration Defaults" here, you can force all Narrations to use certain features such as *Smart Subs, Loop Audio Per Phrase,* or *Single Audio Clip.* This allows you to streamline setting up your Narrations in some cases.
For example, if emulating the repetitive beeping speech audio used in many retro handheld games, you could enable all Narrations to loop audio by default, and thus save yourself from checking the *Loop for Duration* option in each Narration.
Or, if all Narrations have audio and the text is simply acting as subtitles, you could force each Narration to use *Smart Subs*, saving you from having to specify each subtitle's display length.

The NarrationManager's other foldout, "Interactive Settings", contains several fields as well:

- *Button Name:* The name of the button the player will press for any NarrationManager interactions. Note: remember to add/specify this button in Unity's Input Manager.
- *Press to Skip:* The player may press the button to skip through narrations
- *Press to Continue:* The player *must* press the button before the Narration will be considered finished.\
- *Disable Player while Narrating:* The player will be unable to move when Narrations are being played.
- *Character Controller:* The player's character controller (type Monobehaviour) so that it may be disabled when playing. This option only appears if *Disable Player while Narrating* is on.

Code Tips:
The Narration Manager has a singleton design - only one is allowed to exist in any scene. This one instance can be accessed in code via `NarrationManager.instance`.

If you ever want to play a Narration via code, it should be done by calling `NarrationManager.instance.PlayNarration(Narration toPlay)` .

NarrationTriggers

NarrationTriggers are great for when you have a general event or moment that triggers Narrations at several points in your game.

An example of one of these moments is on entering an area - it could be a the beginning area of a new level, a new area on the same map, or the area around an interesting object. Regardless of the circumstance, this is can be generalized to “play a narration when the player enters this area”. This is what the included ProximityNarrationTrigger class does - it asks the NarrationManager to play a Narration when an object enters its area. More on the ProximityNarrationTrigger and other included triggers is given in their respective sections below.

1. ProximityNarrationTriggers -
   This trigger tracks a GameObject’s Transform, refering to it as *Target*. If the *Target* gets within a certain proximity of the trigger, the trigger fires and plays a Narration. It’s other fields are:
   - *Proximity Duration* - How long *Target* must remain in proximity before the Narration will trigger. This is set to zero by default, but can have many uses. For example, if a player sits around in a room for 5 minutes, you might have the Narrator say, “This room isn’t that interesting! How in the world have you managed to occupy yourself for 5 whole minutes here?”
   - *Consecutive* - If *Proximity Duration* is greater than zero, you have the choice of whether or not the time spent in proximity must be consecutive, or if time from different visits and departures adds up to trigger the Narration.
   - *Is 2D* - When checked, the Z-axis is ignored in calculating proximity. Good for 2D games using Unity’s standard 2D orientation.

2. InteractiveNarrationTriggers -
   These are designed to play a Narration when the player nears an object and presses a button to interact with it. The button used is the one specified in the NarrationManagers *Button Name* field.

The Proximity Settings are used to determine whether or not the player is close enough to interact with this object and trigger the Narration. The fields act identically to those explained in the ProximityNarrationTrigger section above.

3. OnEnableNarrationTrigger-
   This one is straightforward - the Narration will trigger when this script is enabled. Take care when allowing this Trigger to be enabled when a Scene first starts - if the trigger runs it's OnEnable() function before the NarrationManager is finished initializing, the NarrationManager may not be prepared to play it.

4. TimerNarrationTrigger-
   This trigger begins counting seconds when it is enabled. When it reaches the time entered in its *Timer* field, it will trigger its Narration.
   Note: These triggers have a "ResetTime()" function which can be called to begin have the timer restart at zero. This can have several uses. For example, let's say the player is collecting boxes. Each time they collect one, the ResetTime() is called and the trigger's timer is reset. However, if they go longer than 10 seconds without collecting a box, a Narration could fire, teasing the player for being so slow.

5. CountingNarrationTrigger-
   This trigger is designed to fire when a certain number is reached, and is used in conjunction with the NarrationCountElement class.
   NarrationCountElements contain a float that other classes can increment or decrement the value of.
   A CountingNarrationTrigger reference a NarrationCountElement and fires when the element's value reaches the value or range prescribed by the trigger's *Relation* and *Target Value* fields.
   For example, we could have a NarrationCountElement that gets incremented every time the player bakes a pie. We could then have a CountingNarrationTrigger that congratulates them when they've baked their tenth pie.

6. CollisionNarrationTriggers -
   This trigger can be added to an object that has either a Collider or a Collider2D. The collider should have isTrigger checked. It also has a list of tags/names called TriggeredBy. If a gameObject (with a collider) who's tag/name is in TriggeredBy touches or enters this collider, the Narration will be triggered.
   Note: Remember, in Unity collision detection between 2 objects generally requires at least one of the objects to have a Rigidbody (or Rigidbody2D).

NarrationTriggers also have some added features which might come in useful:

- Use A NarrationList (toggle option): Instead of playing it's built-in Narration, you can have a Narration play from a NarrationList

- Reactions: Depending on whether or not the NarrationManager succeeded in playing or queuing its Narration, a NarrationTrigger can react in several different ways. These ways include:
    - *Disable*: disables this NarrationTrigger component
    - *Disable Gameobject*: disables the Gameobject this NarrationTrigger component is on.
    - *Destroy*: Destroys this NarrationTrigger component
    - *Destroy Gameobject*: destroys the Gameobject this NarrationTrigger component is on.
    - *Reset*: Will reset the trigger after "Reset Wait" seconds have passed so that it can be fired again.

A NarrationTrigger has an OnSuccess reaction and an OnFailure reaction. OnSuccess is used if the Narration was either immediately played OR added to the NarrationManager's queue. Otherwise, OnFailure is used.

This distinction between failure and success can be useful. For example, maybe on entering a beautiful area, you have a ProximityTrigger that has the Narrator to say something poetic. But what if a different Narration is already being played? You don't want to queue the line because if the player leaves the area before it's played, it won't make as much sense. But you also don't want this line of narration to just go to waste...

The solution is to set OnFailure to "Reset", and then set the Reset Wait to something reasonable - maybe that's 20 seconds, or maybe it's 5 minutes - it could depend on the map. After that amount of time has passed, the ProximityTrigger will be reset. If the player re-enters the area, it will try to play its Narration again. This time, the NarrationManager hopefully won't be busy.

Coding Tips:

Important: When making your own extensions of the NarrationTrigger class, if you include an `OnDisable()` function, be sure to call `base.Disabled()` at the end of it. If you include an `OnDestroy()` function, be sure to call `base.Destroyed()` at the end of it.

Also, remember that sometimes it might be easier to avoid using NarrationTriggers and simply hardcoding a Narration into one of your scripts. This could happen if you have a simple case that doesn't fit any of the provided triggers and only occurs once or twice.

NarrationLists

A NarrationList is basically a pool, or list, of Narrations. Instead of playing a specific Narration, a Narration Trigger could play a Narration from a NarrationList. The NarrationList decides which of its Narrations to play when a Trigger asks it to play. NarrationLists allow for several neat tricks and benefits:

- *Looping* - Maybe you have 100 special collectibles in the game, and whenever the player gets one, the narrator says something clever. However, let's say you only have 8 different Narrations that go with this event. By putting these 8 in a NarrationList and enabling looping, the Narrator will go through the pool repeatedly without you having to set up a separate Narration for each and every collectible.

- *Randomization* - Using the looping example, maybe you don't want the Narrations in the NarrationList to play in the exact same order everytime - it's just too predictable. You can check randomization in the list to add some variety.

- *Play in Order* - Let's say the player has to turn on 3 light switches, and they're free to do so in any order. When they reach the first one, the Narrator says "That's one." On the second, the Narrator says, "You're getting the hang of this." On the third, "Finally. It's about time!" Using plain Narrations won't work in this case - we can't predict which order the switches will be activated in. However, if we put the 3 Narrations, in order, into a NarrationList, each switch can draw from there instead when the player turns it on. The first switch turned on will get the first Narration in the list, the second switch gets the second Narration, and the third will get the third Narration.

These are just a few possible design points and patterns where a NarrationList might be useful.
NarrationLists CAN hold empty narrations. So, as in the *looping* section above, if you don't want a Narration to play every single time a collectible is collected, put a few empty Narrations in the NarrationList. If the NarrationList choose an empty entry to play, nothing will happen. Just silence.

Code Tip: If you have a reference to a NarrationList, you can ask it to play via `.Play()`

The Subtitle Manager

The Subtitle Manager is in charge of, you guessed it, displaying subtitles. It displays the current subtitle on the display canvas, gradually scrolls through the subtitle if it's too large to display all at once, and then hides the display panel when there are no subtitles to read.

The NarrationSystem prefab has a SubtitleManager prefab already built and childed to it. It is best to begin with this prefab and simply change parts of it, rather than trying to start from scratch.

Primary Fields/Properties:

Font Settings
Font Size Modifier: The percentage of the display area's height a line of text should take up. (Example: 25 would mean 25%. So 4 lines could be displayed at once).

Font Size Range: Minimum and Maximum font sizes

User Prefs Settings
PrefsKey: The PrefsKey is the string that will be used to store/index whether or not the subtitles are on in Unity's PlayerPrefs.

On By Default: If there is no PlayerPrefs information (such as when the player starts a new game) the subtitles will be automatically turned on.

Other:
Time Padding: How long after the audio is finished should the subtitle display remain open for.

On In Editor: This is for toggling the subtitle visibility when in the scene editor.

Typing Animation: Text will appear letter by letter, as if being typed.
Max Time Between Letters: If Typing Animation is enabled, you can specify the max amount of time between letters popping up. If a subtitle is too long to be fully typed out at this speed before the Narration's duration is up, the time between letters will automatically be scaled down so that the full subtitle can be displayed within the allotted time.

Customizing UI Art:
If you want to customize the appearance of your subtitle display, begin by looking the SubtitleDisplay, which is a child of the SubtitleManager. All subtitle UI components belong to the SubtitleDisplay or one of its children.

Altering the Font and Color

Navigating the heirarchy, find the TextUI gameObject via SubtitleDisplay-> ScrollingArea-> TextUI. TextUI has the Text component used for drawing subtitles. Change the color and font as desired.

Altering the Display's Image:

If you have a sprite you'd like to use for the background image instead of the default black box look at the SubtitleDisplay's Image component. Place the sprite you'd like in the component's "Source Image" field. Then alter the component's Color field as desired.

If you don't have a sprite but do want a differently colored box, simply change the Color field.

Resizing the Display:

The SubtitleDisplay is a standard Unity UI Panel, and thus you can resize and re-anchor the display as you see fit. Here are some tips:

- For even scaling across various screen sizes, it's good to keep SubtitleDisplay's anchors at its own corners
- ScrollingArea, a child of SubtitleDisplay, is the area where the text is displayed. You can resize it as well. We recommend you keep it slightly smaller than the SubtitleDisplay, and that the ScrollingArea's anchors be at its own corners.

Overlays

You can draw things over the SubtitleDisplay panel and the text by making them children of Overlays, or by making them the last children of SubtitleDisplay in the heirarchy (latter UI elements are drawn on top of the previous by default in Unity). This is how the fade-in-fade-out of text in the SubtitleManagerPrefab is accomplished.

Coding Tips:

The NarrationManger generally tells the subtitle what to play and when. However, if you find yourself needing to ask the manager to display a subtitle from elsewhere in code, note that displaying a subtitle has two aspects or parts:

1. The integer value returned by `PlayerPrefs.GetInt(PrefsKey)` must be 1. This means that subtitles have been enabled.
2. The `DisplaySubtitle(string subtitle)` function must be called on an instance of the Subtitle Manager. `NarrationManagers` do this when playing a Narration that has subtitling.

Honorable Mention: NarrationChains

NarrationChains aren't as major as most components in the Narrate system, but they can be useful, particularly in creating conversations and opening up new dialogues or narrations.

NarrationChains allow you to specify an initial NarrationTrigger, referred to as the *Head.* You may then begin to add links to the chain, each containing another NarrationTrigger.

When the previous NarrationTrigger in the chain is either disabled or destroyed (each link specifies), the next will be enabled.

So, say we have an InteractionTrigger. We could set it's *OnSuccess* reaction to "Disable", so that it disables itself after successfully playing its Narration. This could trigger a new InteractionTrigger to be enabled. Now the player can interact again, with something entirely different being said.
This same functionality could in some cases be accomplished with an InteractionTrigger which draw from a NarrationList and has its reaction's set to "Reset", but in some instances, you may find chains to be cleaner. Look at the 2D demo's lobster character for an example of this.

Support

Got a question? A bug? Or maybe you have a feature you're not sure quite sure how to implement. Email:

dev.glassed@gmail.com