ABLEANIMATOR

MugHeadStudios

OVERVIEW

*Note that this documentation only details what is necessary for interfacing with the package. If you have any further questions, please contact me, or check the **.cs** files themselves which have been heavily commented.*

What is AbleAnimator?

AbleAnimator is a powerful and easy to use animation package for Unity that enables you to quickly and easily animate any game object in your scene, including characters, props, and special effects. AbleAnimator also provides a number of handy features, such as key frame animations that can control the sprite, position, scale or rotation of an object, or even run events.

Table of contents

Please note if you see any mistakes, please contact me using one of the options at the end of this document.

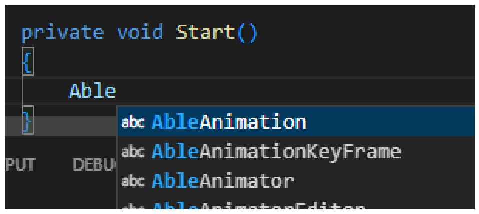
DOCUMENTATION

1. Prerequisites (Accessing AbleAnimator in code)

When working with AbleAnimator it is recommended to use the MugHeadStudios namespace for quick access to the classes.

```
using MugHeadStudios.AbleAnimator;
```

Place the above within the list of namespaces at the top of your .cs file, now you can access the classes / instances directly:

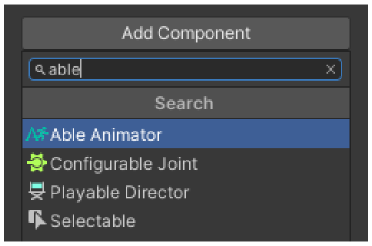


```
private void Start()
{
    Able
}
```

2. AbleAnimator in the editor

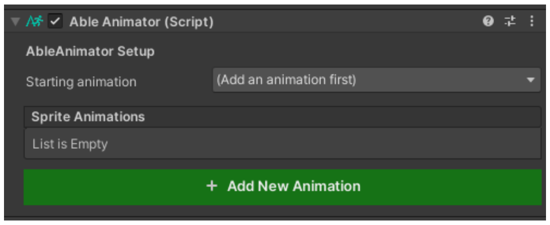
2a. Add AbleAnimator to a game object

To add the AbleAnimator component to a game object, simply select the game object, click the Add Component button in the Inspector then search and select the Able Animator component.



2b. Add a new animation

To add a new animation, click the large green “Add New Animation” button. On adding an animation you will notice 3 common fields you can modify, the alias, the mode, and the play amount. The alias is how we will reference this new animation, the mode can wither be “Sprite Mode” or “Key Frame Mode” which are explained in 2b. and 2.c, and finally the play amount is how many times this animation should be played.


Able Animator (Script)

AbleAnimator Setup

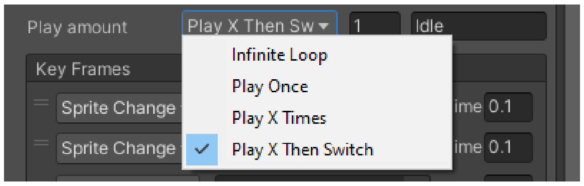
Starting animation (Add an animation first)

Sprite Animations

List is Empty

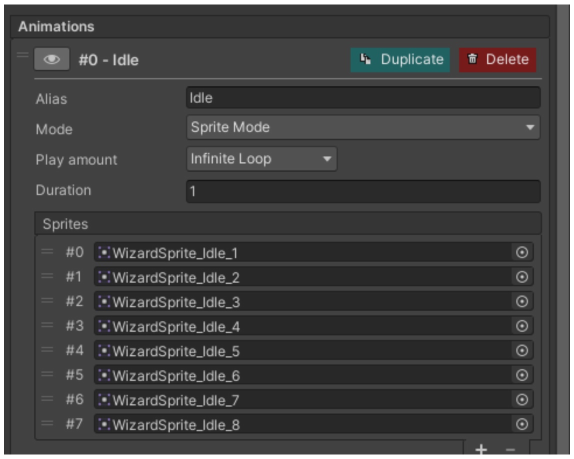
2c. Play amount

The first two options should explain themselves. “Play X Times” takes an integer, where -1 is infinite, and any positive number will play that amount of times. “Play X Then Switch” also takes a string for an animation to switch to on completion.


Play amount Play X Then Sw 1 Idle

2d. Sprite Mode

Sprite Mode allows you to add any number of sprites to the sprite list which will animation for a specified duration, and either loop or stop depending on the play amount setting. This is perfect for an animation with a consistent speed, but delays can be added, if necessary, by simply adding multiple frames of the same type to the sprites list.


Animations

#0 - Idle

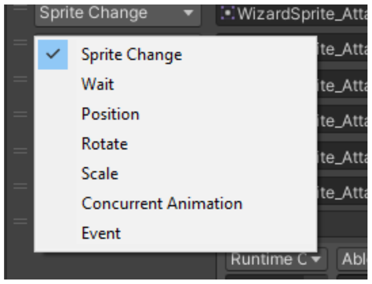
Alias	Idle
Mode	Sprite Mode
Play amount	Infinite Loop
Duration	1

Sprites

#0	WizardSprite_Idle_1
#1	WizardSprite_Idle_2
#2	WizardSprite_Idle_3
#3	WizardSprite_Idle_4
#4	WizardSprite_Idle_5
#5	WizardSprite_Idle_6
#6	WizardSprite_Idle_7
#7	WizardSprite_Idle_8

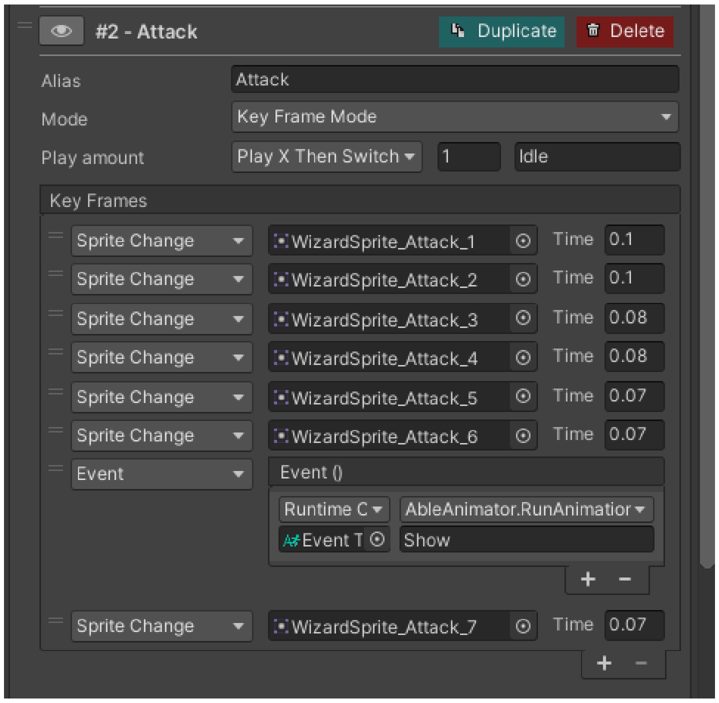
2e. Key Frame Mode

Key Frame Mode is far more versatile than Sprite Mode, notice first that we have a few different options for each key frame:



- Sprite Change – Changes the current sprite and waits for the specified time
- Wait – Simply waits for the specified time
- Position – Moves the game object towards a position, either relative or absolutely, over the specified time
- Rotate – Rotates the game object towards a rotation, either relative or absolutely, over the specified time
- Scale – Scales the game object towards a size, either relative or absolutely, over the specified time
- Concurrent Animation – Does one of the following (Note: concurrent animations do not affect the time of the parent):
    - Run – Runs a new concurrent animation by alias (If a concurrent animation is already running then overwrite it)
    - Append – Appends a new concurrent animation
    - Stop – Stops a running concurrent animation by alias
    - Stop all – Stops all running concurrent animations
- Event – Runs any number of specified Unity Events

An example of a Key Frame Mode animation below:



Alias: Attack
Mode: Key Frame Mode
Play amount: Play X Then Switch | 1 | Idle

Key Frames

Sprite Change	WizardSprite_Attack_1	Time	0.1
Sprite Change	WizardSprite_Attack_2	Time	0.1
Sprite Change	WizardSprite_Attack_3	Time	0.08
Sprite Change	WizardSprite_Attack_4	Time	0.08
Sprite Change	WizardSprite_Attack_5	Time	0.07
Sprite Change	WizardSprite_Attack_6	Time	0.07
Event	Event ()		
	Runtime C / AbleAnimator.RunAnimatior		
	Event T / Show		
Sprite Change	WizardSprite_Attack_7	Time	0.07

2f. Animation interface

Once an animation has been added, notice that to the left of the alias at the top left is a button with the icon of an eye, this button allows us to toggle the collapsing of this animation in case we have many animations attached to this game object so we can see the list of animations easier. The duplicate button will make a copy of this animation and add it below the current index, and finally delete will completely remove this animation from the list of animations.

Notice that there are 2 modes. The first and simplest is “Sprite Mode” which the “Key Frame Mode” accepts a number of key frames which can do a large number of things, but this handles its own timing but waiting a specified amount of time per keyframe before moving to the next, this allows us to have more control over the animation, and even run events or combine animations together.

3. Classes

3a. AbleAnimator Class

AbleAnimator is a class that provides functionality for animating 2D sprites in Unity. It has a list of animations that can be played, and provides methods for playing, stopping, and managing those animations. It also has a SpriteRenderer component which is used to render the sprites for the animations.

Public fields:

List<AbleAnimation> **ableAnimations** - A list of AbleAnimation objects. These are the animations that can be played by this AbleAnimator.

int **startingAnimationIndex** - The index of the animation that should be played when this AbleAnimator is started.

AbleAnimation **currentAnimation** - The animation that is currently playing.

List<AbleAnimation> **concurrentAnimations** - A list of animations that are playing concurrently.

SpriteRenderer **spriteRenderer** - The SpriteRenderer component that is attached to this GameObject.

Public Methods:

AbleAnimation **AddAnimation**(AbleAnimation **animation** = null) - Adds an animation to the list of animations.

void **RunAnimation**(string **alias**) - Runs the animation with the specified alias.

void **RunAnimation**(int **index**) - Runs the animation with the specified index.

void **RunConcurrentAnimation**(string **alias**) - Runs the concurrent animation with the specified alias.

void **RunConcurrentAnimation**(int **index**) - Runs the concurrent animation with the specified index.

bool **IsAnimationPlaying**(string **alias**) - Returns true if the animation with the specified alias is playing, false otherwise.

bool **IsAnimationPlaying**(int **index**) - Returns true if the animation with the specified index is playing, false otherwise.

bool **IsConcurrentAnimationPlaying**(string **alias**) - Returns true if the concurrent animation with the specified alias is playing, false otherwise.

bool **IsConcurrentAnimationPlaying**(int **index**) - Returns true if the concurrent animation with the specified index is playing, false otherwise.

void **StopAllAnimations**() - Stops all animations.

void **StopConcurrentAnimations**() - Stops all concurrent animations.

Private Methods:

void **Start**() - MonoBehaviour method, called at the start of the scene

void **Update**() - MonoBehaviour method, called each frame

3b. AbleAnimation Class

The AbleAnimation class is used to manage animations for the AbleAnimator class. It has variables and functions for managing animation playback, including setting the animation's duration, mode (sprite or keyframe), play amount, and onComplete action. It also has an Update function which is called every frame to update the animation.

Public Fields:

AbleAnimator **parent** - The parent AbleAnimator that this animation is a child of.

Action<AbleAnimation> **onComplete** - An action to run once this animation has completed. Note that if play amount is infinite (-1) this will never run.

PlayAmountMode **playAmountMode** - The mode to determine how many times this animation should play.

int **playAmount** - Play amount, -1 == infinite, 0 == none, any positive number will play that number of times.

string **switchOnComplete** - The name of the animation to switch to once this animation has completed.

int **currentPlayCount** - The number of times this animation has played.

bool **editorCollapsed** - Whether this animation is collapsed in the editor.

string **alias** - The name of this animation.

float **duration** - The duration of this animation in seconds.

Modes **mode** - The mode that this animation is in, either SpriteMode or KeyFrameMode.

List<Sprite> **sprites** - A list of sprites to use for this animation in SpriteMode.

List<AbleAnimationKeyFrame> **keyFrames** - A list of keyframes to use for this animation in KeyFrameMode.

Private Fields:

int **currentFrameIndex** - The index of the current frame in the animation.

float **frameWaitTimer** - A timer to track how long to wait until the next frame in the animation.

Vector3? **origionalLerpVector** - The vector to use for Lerping in KeyFrameMode.

float **frameTimer** - A timer to track how long the current frame has been active.

Public Methods:

AbleAnimation **Run**(AbleAnimator **ableAnimator**, bool **forceRestart** = false) - This method sets the animation's parent to the provided AbleAnimator, and sets the current frame index to 0. If forceRestart is true, or the animation's current animation is not this animation, the frameTimer and frameWaitTimer are reset to 0, the origionalLerpVector is set to null, the currentPlayCount is set to 0, and stopped is set to false. This method returns the animation.

AbleAnimation **SetFrameIndex**(int **newFrameIndex**) - This method sets the current frame index to the provided newFrameIndex. The frameTimer is reset to 0. This method returns the animation.

AbleAnimation **Stop**() - This method sets stopped to true. This method returns the animation.

void **Update**() - This method updates the animation. If stopped is true, the method returns. Otherwise, if the mode is equal to Modes.SpriteMode, SpriteUpdate is called. Otherwise, if the mode is equal to Modes.KeyFrameMode, KeyFrameUpdate is called.

Private Methods:

void **SpriteUpdate**() - Responsible for animating the sprite across frames. It does this by incrementing the current frame index every time the frame timer exceeds the frame time. If the current frame index exceeds the sprite count, it will reset back to 0.

void **KeyFrameUpdate**() - Responsible for animating the object across frames in different ways depending on the current keyFrameType. It does this by incrementing the current frame index every time the frame timer exceeds the frame time. If the current frame index exeeds the number of key frames, it will reset back to 0.

Vector3 **KeyFrameVectorLerp**(AbleAnimationKeyFrame **currentKeyFrame**) - This method is responsible for interpolating the position of an object between two keyframes in a timeline. The first keyframe is the current keyframe, and the second keyframe is the next keyframe in the timeline. The method takes in the current keyframe, the object's original vector position, and a boolean value indicating whether the object's position is relative to the timeline or not. If the position is relative, the object's position will be offset by the position of the first keyframe. The method returns the new interpolated position of the object.

3c. AbleAnimationKeyFrame Class

The `AbleAnimationKeyFrame` class is a data structure used to represent a key frame in an animation. It contains information about the type of key frame, the sprite to use (if applicable), the position, rotation, and scale (if applicable), whether the key frame is relative or not, the time at which the key frame occurs, and any additional action to take (if applicable).

Enums:

enum **KeyFrameType** - Used to determine what type of keyframe is being used. There are four different types of keyframes: sprite change, wait, position, and rotate.

enum **ConcurrentAnimationType** - Used to determine how concurrent animations should be handled. There are four different options: run, append, stop, or stop all.

Public Fields:

KeyFrameType **keyFrameType** - The type of KeyFrame that can be used by AbleAnimation

Sprite **sprite** - The sprite to swap to if the keyFrameType is set to "SpriteChange"

Vector3 **vector3** - Used for keyFrameType's "Position", "Rotate", or "Scale"

bool **relative** - Used with vector3 to determine whether the value is relative or absolute

float **time** - Used alongside keyFrameType's "Wait", "Position", "Rotate", or "Scale"

string **stringValue** - Used with KeyFrameType "ConcurrentAnimation"

ConcurrentAnimationType **concurrentAnimationType** - Which concurrent animation type

Action **action** - The event that will be invoked if keyFrameType is set to "Event"

3d. AbleAnimatorEditor Class

The AbleAnimatorEditor class is a custom editor for the AbleAnimator class that provides a way to add, edit and remove animations attached to a game object. The editor controls how the component is displayed within the inspector, and provides the user with full control of the animations and comes with many handy features that helps the user be as efficient as possible when handling animations for your game object and scene. Because the editor class itself is quite large and is not meant to be interfaced with directly by the user, please check the .cs file itself as it is heavily commented. Otherwise if you have any further questions please contact me.

4. Conclusion

Thank you for using MugHeadStudios AbleAnimator, if you have any suggestions, notice any mistakes within the documentation, or bugs in the package, please let me know as it is very much appreciated.

5. Contact

Craig Dennis (Zephni)
MugHeadStudios

Contact me by email: zephni@hotmail.co.uk

Discord: https://discord.gg/S7Y3RyUpGh