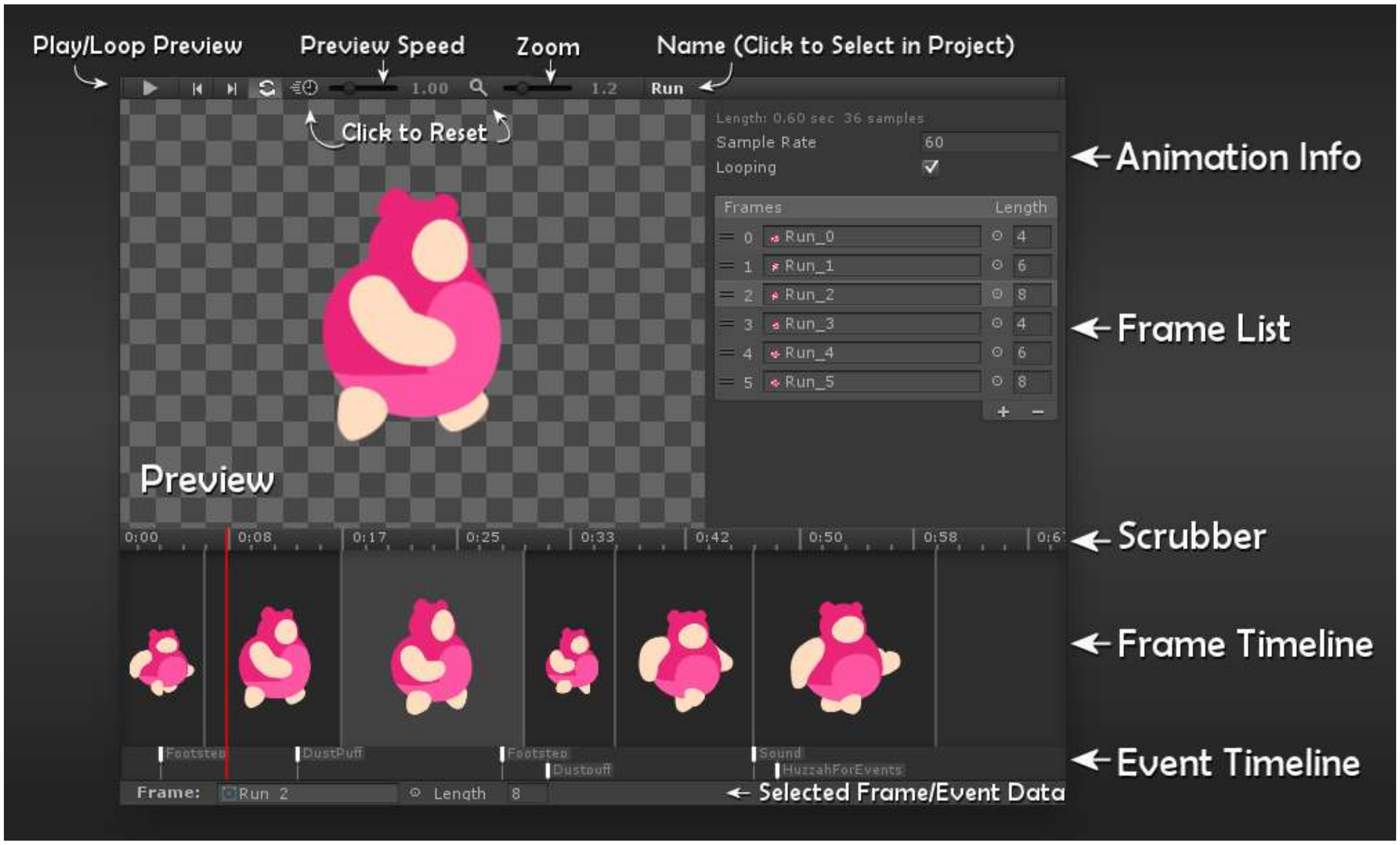
PowerSprite Animator

Version 1.2

Overview



Length: 0.60 sec 36 samples

Sample Rate 60

Looping ✓

	Frames	Length
0	Run_0	4
1	Run_1	6
2	Run_2	8
3	Run_3	4
4	Run_4	6
5	Run_5	8

Frame: Run_2 Length 8

[Editor Overview Video (Youtube)]() | [Scripting Example Video (Youtube)]() | [Unity Forum Thread]()

Quick Start

- **Open editor** by clicking Window -> PowerSprite Animator on the taskbar
- **Create a new animation** in the normal way you do in unity. I click Create ->Animation in the project.
- **Insert** on frames by dragging sprites over the timeline
- **Delete** sprites/events by selecting them and pressing delete on keyboard

The editor is designed to be super simple to use, drag and drop interface, you shouldn't have too much trouble!

Keyboard Shortcuts

- **Space** – Toggle Play/Pause
- **Left/Right arrows**, select next/previous frame
- **Delete** – Delete selected frames
- **Ctrl-A** – Select All frames (if an event is selected it will select all events)
- **Ctrl-D** – Duplicate selected Frame/Events
- **Ctrl-C** – Copy selected Frames/Events
- **Ctrl-V** – Paste Frames/Events on timeline

Opening an animation

- Open the Animator- On the taskbar click Window -> PowerSprite Animator.
- Create a new animation by right clicking in the project -> Create -> Animation
- Or, find an existing animation in your project window and select it

Editing an Animation

- **Insert Frames**- To insert frames, simply drag one or more sprites from the project window to the timeline. A blue bar indicates where they will be inserted in an existing animation.
- **Zoom/Pan-** Use the mouse wheel to zoom, and right or middle mouse button to pan around either the Preview or the Timeline.
- **Select Frames**- Click on a frame in the timeline, or click and drag to select multiple frames.
  Hold Ctrl or Shift to select multiple frames/Events. Ctrl-A selects all.
- **Delete Frames**- Select frames, and press delete on your keyboard. Alternatively, select frames in the lit, and click the – symbol
- **Re-order frames**- Select one or more frames then drag them to reorder, new position is indicated with a blue bar.
- **Frame duration**- Click and drag the line between frames to shorten/lengthen. Alternatively type in the desired length in the list view. If multiple frames are selected, they can be resized together.
- **Animation Info**:
  - **Sample Rate**- Frames and event times are snapped to the current sample rate, increase this for greater flexibility. It's equivalent to "Frame Rate" in unity's animation editor.
  - **Length (sec)**- Quickly speed up/slow down your animation. This option adjusts the animation's frame rate to get your animation to match a certain length in seconds.
  - **Looping** – Whether the animation is set to loop
  - **Animated Sprite Type**- Choose whether the animation is for a SpriteRenderer or a UI Image

Animation Events

- Animation Events are messages that unity sends to a game object at a certain time in an animation. They're super useful for things like playing sounds, spawning effects, or synchronizing things with your animations.
- Events live on the event timeline below the frame timeline.
- Creating Events
  a. Double click on event bar to create an event.
  b. Type the function name of the event- This function will be called in a component on the game object that plays the animation.
  c. Select optional parameter from the dropdown box.
- Event options:
  a. **Require Receiver**- if ticked, you will get an error if the event isn't received (if there's no function of that name in a component of the object playing the animation)
  b. **Add 'Anim' Prefix**- This is handy if you like to have a standard prefix on event functions so that you can tell easily in where they are called from. If ticked, you need to add "Anim" to the start of your function. (Eg, an event called PlaySound requires a function called void AnimPlaySound())
  c. **Send Upwards** (EXPERIMENTAL)- Usually an event message is sent to the GameObject that's playing the animation. Sometimes it's nice to have the animating sprite be a child of the object that has all the scripts.

     If this is ticked, the event will be sent upwards from the sprite objects to its parent game object. This requires either the sprite to have a SpriteAnim, or SpriteAnimEventHandler component. (it's also less efficient than a normal animation event)

ANIMATION NODES

- Animation Nodes let you define positional points on frames of your animation.
- Use them to attach things to your sprites!
- **Editing nodes:**
  a. **Double click** on the preview to add a node, **drag it around** to position it.
  b. You can **rotate** the selected node too if it needs an angle. (Hold Ctrl/command to snap)
  c. TIP: When doing a lot of frames use the left/right arrow keys to skip through frames while you tweak the positions.
- **Reading node position/angle**
  a. Add the SpriteAnimNodes component to your animated sprite object
  b. In the scripts you get a nodes position or rotation by its ID (there are up to 10)
  c. There's really just 2 functions to use:

```
/// Returns the position in world space of the specified node
public Vector3 GetPosition(int nodeId)

/// Returns the rotation angle in world space of the specified node
public float GetAngle(int nodeId)
```

- **Example scripts:**
  a. Here's a quick example script to attach a child object (a hat) to node 0:

```
void LateUpdate()
{
    SpriteAnimNodes nodes = GetComponent<SpriteAnimNodes>();
    transform.FindChild("Hat").position = nodes.GetPosition(0);
}
```

  b. And here's an example of a function that also sets rotation, and handles the character being flipped:

```
void LateUpdate()
{
    // Update 3 attachments
    UpdateAttachment( transform.GetChild("Hat"), 0 );
    UpdateAttachment( transform.GetChild("Gun"), 1 );
    UpdateAttachment( transform.GetChild("Moustache"), 2 );
}

// Updates the position/angle of an attached transform to a node.
void UpdateAttachment( Transform attachment, int nodeId )
{
    SpriteAnimNodes nodes = GetComponent<SpriteAnimNodes>();
    attachment.position = nodes.GetPosition(nodeId);
    float angle = nodes.GetAngle(nodeId);
    // If this object is flipped, then un-flip the angle to compensate
    if ( transform.lossyScale.x < 0 )
        angle -= 180.0f;
    attachment.eulerAngles = new Vector3(0,0,angle);
}
```

SpriteAnim Component

- *This is optional*, you can use the PowerSprite Animation editor without it.
- This component lets you play animations from script without the need to create a Mechanim Animator for each object and State for each animation.
- Check out this video for an example: https://www.youtube.com/watch?v=a2EzLTAym7U
- It's useful when the Animator is overkill and you just want to handle animating things in your scripts.
- Here are the basic functions it supports, and there's an example of use on the following page.

```
/// Plays the specified clip
public void Play(AnimationClip anim)

// Stops the clip by disabling the animator
public void Stop()

/// Pauses the animation. Call Resume to start again
public void Pause()

/// Resumes animation playback at previous speed
public void Resume()

/// True if an animation is currently playing (even if paused)
public bool Playing { get; }

/// Property for pausing or resuming the currently playing animation
public bool Paused { get; set; }

/// Property for setting/getting the current playback time of the animation
public float Time{ get; set; }

/// Property to get or set the the normalized time (between 0.0 to 1.0 from star
t to end of anim) of the currently playing clip
public float NormalizedTime { get; set; }

/// The currently playing animation clip
public AnimationClip Clip { get; }

/// The name of the currently playing animation clip
public string ClipName  { get; }
```

- Here’s an example of a simple component that plays an animation and destroys itself. Pretty basic stuff, but you get the idea…

  To use it, you’d create a game object, add the Effect component, and drag an animation over “Animation” field in the inspector.

```
using UnityEngine;
using System.Collections;
using PowerTools;

/// Plays an animation and then destroys itself
[RequireComponent(typeof(SpriteRenderer))]
[RequireComponent(typeof(SpriteAnim))]
public class Effect : MonoBehaviour
{
    [SerializeField, Tooltip("The animation to play!")]
    AnimationClip m_animation = null;

    SpriteAnim m_spriteAnim = null;

    // Use this for initialization
    void Start ()
    {
        // Store the component so we don't have to get it every frame
        m_spriteAnim = GetComponent<SpriteAnim>();

        // Play the animation
        m_spriteAnim.Play(m_animation);
    }

    // Update is called once per frame
    void Update ()
    {
        // If the animation has finished playing, destroy the object
        if ( m_spriteAnim.Playing == false )
            Destroy(gameObject);
    }
}
```

Version History

- **v1.2**
  - Added animation nodes functionality
  - New functionality in SpriteAnim component:
    - Set current animation time.
    - Set animation playback speed**.**
    - Pause and resume animations.
  - Fixes to support Unity 5.6
- **v1.1**
  - Added support for UI Image animations
  - Can now Copy/Paste/Duplicate selected Frames and Events (with Ctrl-C,V,D respectively)
  - Added ability to select multiple Frames/Events by holding Ctrl or Shift when selecting
  - Added ability to "Select All" with Ctrl-A
  - Added ability to move back/forward through frames with left/right arrow keys
  - Events now reposition when editing frames to stay synced to corresponding frame
  - Added space at start of timeline to make it easier to click things at the beginning of an animation
  - Added editable Length field to allow quick adjustments to the animation speed (by adjusting the frame rate)
  - When sprites are dragged to the timeline, they are now sorted into natural order (eg: Walk9.png now comes before Walk10.png). Thanks to Fausto Cheder for this one!
  - Minor fixes to support Unity 5.5
- **v1.0**
  - Initial Release

Support

Unity Forums Thread - https://forum.unity3d.com/...

Email Dave - support@powerhoof.com

Credits

Made by Dave Lloyd @duzzondrums to help Barney @powerhoof add more sweet animations to our games!

Check out our website to see the results: http://www.powerhoof.com

Demo sprites by Barney Cumming, from Regular Human Basketball

Thanks also to ex2d which we used for aaaages, and which inspired some of the layout.