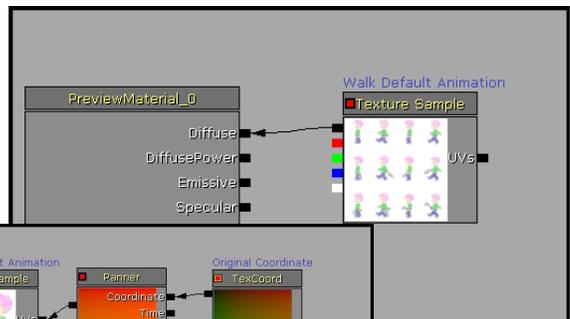


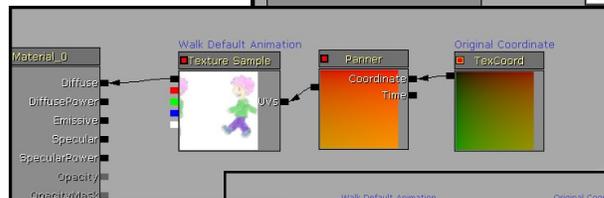
## Animate with a spritesheet in the material editor.

I developed this animation system when prototyping for a personal project. I used simple 2d animations with two to four frames each and played them onto a 3D object that also had its own animations. Note: This material may only work well if the sprite sheet has an even total number of frames, such as 2,4,6,8 and so on. This material is also only made for looping animations in this tutorial and wont stop playing.

Add the sprite sheet to a **Texture Sample**, it maybe a good idea to convert this to a Texture2D parameter as we build this material to be used with instances and may want to use different sprite sheets when we are done.

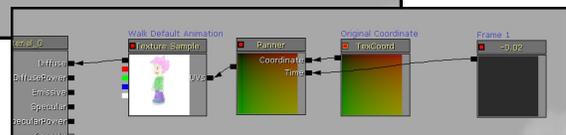


Add one **Panner** and set it to pan 1 in X and 0 in Y. Attach it to the texture uv.



Add a **Texture Coordinate**.

The texture is going to pan in X like crazy now.



Add a **Constant**, connect it to the Panner Time.

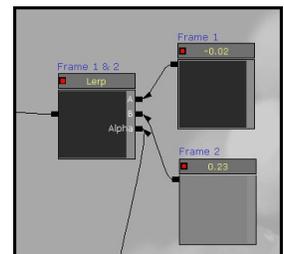
Now it's possible to set the Texture Coordinate to fit the frame row.

Center the first frame by adding a value to the constant. Add more constants and setup their values to fit each frame of the animation.

Take the constants that represent frame one and two and connect them to a **Linear Interpolation(Lerp)**.

Frame 1 into A and Frame 2 into B. For the Alpha, make a temporary **Constant** and plug it in for now.

Connect this linear interpolation to the Panner's Time.



Next Page:

Making the animation play.

This tutorial will only show how to make this work with a four frame long animation. Should be possible to extend this to play longer animations as well.

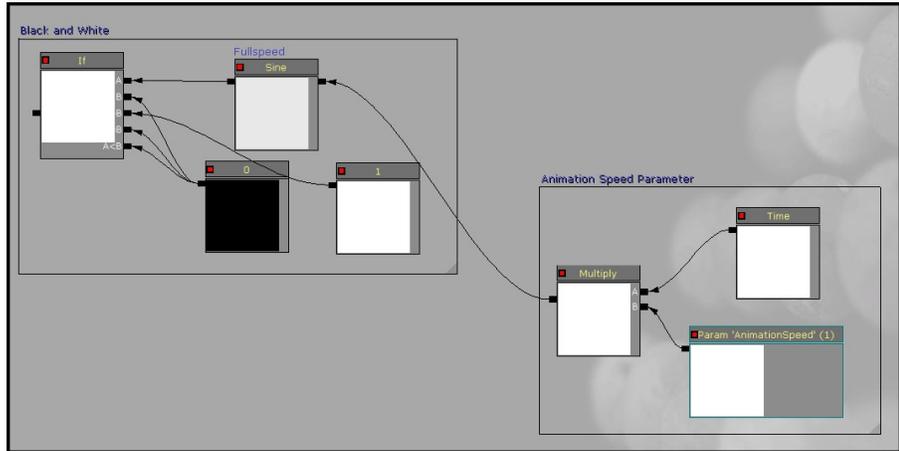
## Animation system

We would like to switch between the frame numbers over time. Its possible to switch between black(0) and white(1) with a Linear Interpolation but we wouldnt like to blend from black to white with all grey middle values. Thats why an IF is necessary to only make our output black or white.

Add **Time**, **Multiply** and a **Scalar parameter**.

Connect Time into Multiply A and the Parameter into B.

This will let us control the animation speed.

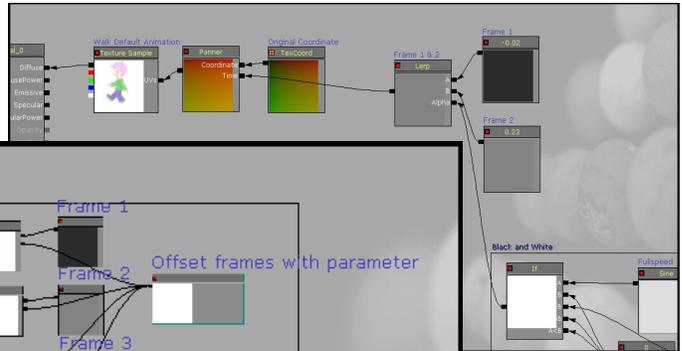


Add **IF**, **Sine** and two **Constants** to switch between 0 and 1.

Connect the animation speed into the Sine, making it go from black to white over time. ( Sine value = 1 ) Connect the Sine into IF A and set the IF to only output 1 or 0 from it, like this: **B is 0, IF A is equal or less then B output 0, IF A is greater then B output 1.**

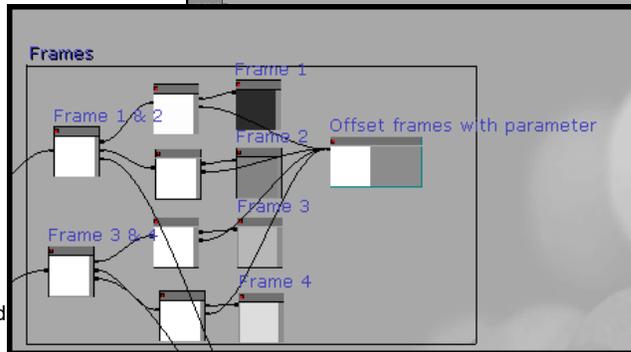
Connect this "animation system" to the **Linear Interpolation's** alpha.

These two frames are now going to loop and its possible to set the animation speed.



We would probably like to have more then two frames in the animation though. Maybe it also would be nice being able to offset all our frame positions with a scalar parameter.

Add one **Add** node for each frame and one **Scalar parameter**. Connect frame to add A and add to Linear Interpolation. Connect the Scalar



Parameter to all of the add nodes in B. We can now offset all frames with one value.

Reconnect all frames to the Linear Interpolation. Make a new **Linear Interpolation(Lerp)** for frame 3 and 4.

Connect the "animation system" to these two linear interpolations Alpha.

If we like more then two frames to loop, we would need to loop the Linear Interpolation nodes at half speed. Let's set up a new part to the "animation system" that will let us have four frames.

**Duplicate the IF, Sine and two constants.**

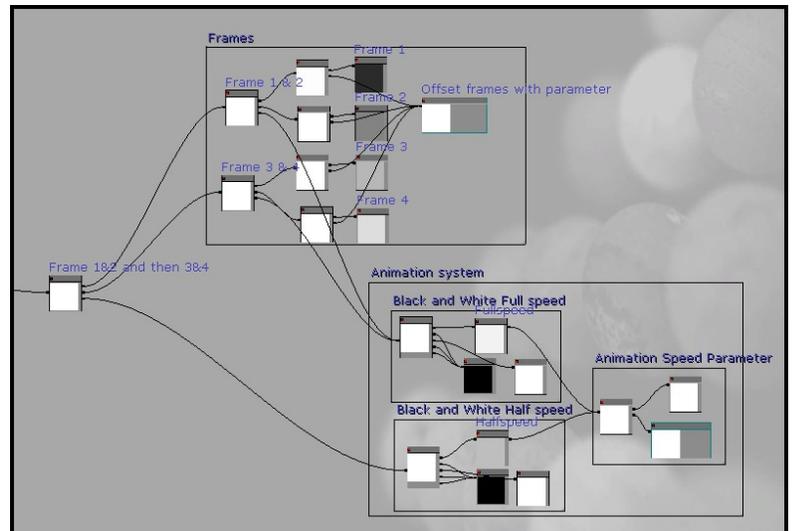
Set Sine value 2 and connect the "animation speed" to it.

Add another **Linear Interpolation(Lerp)**.

Connect the other two linear interpolations into A and B.

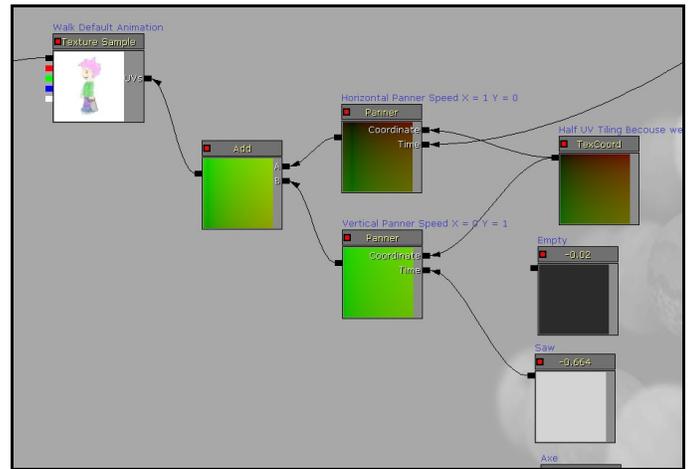
Connect the new part of the "animation system" into the alpha, making it play frame 1 & 2 and then frame 3 & 4.

Next page: Change height position to play a different animation.



Switch to another row in order to play another animation from the same sheet. Instead of making the position change in X (Horizontal) like we did with the animation frames make them change in Y (vertical) instead. First we need a new Panner to control the height position separate from X and then use switch parameters to change between the rows using instance materials or through unrealscript.

- Add a new **Panner**, set its Y to 1 and X to 0.
- Add a new **Add** and connect our old and new Panner to it.
- Connect this to our Texture's uv.
- It will also pan like crazy in Y until we add a new **Constant** to it.

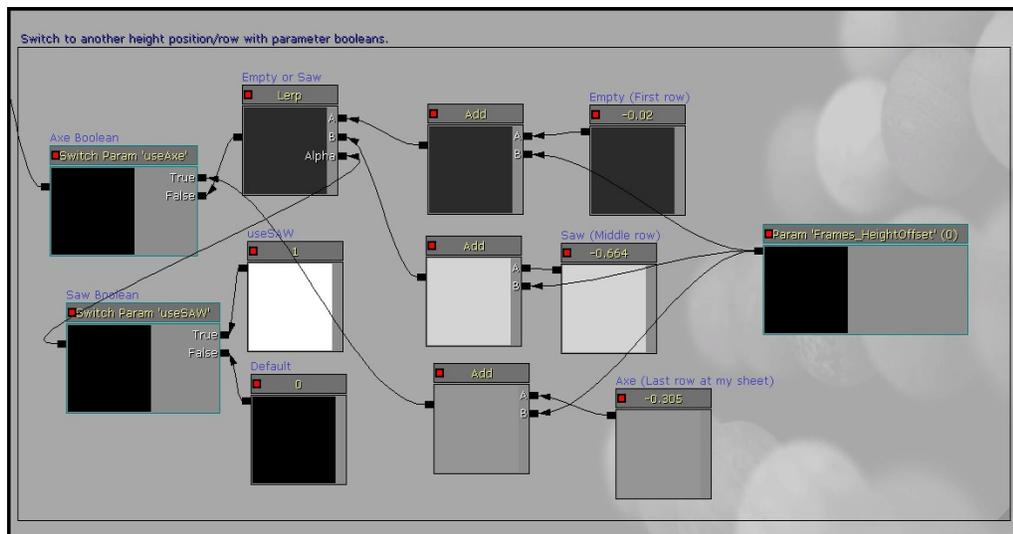


Our Texture Coordinate will be doubled, we have to reduce it to half the size in U and V to be the same as before, now when we're using a second Panner.

Switch between height positions is almost the same as with our frames, we have the row (frame) position and make one row position for each animation in the sheet.

Add a **Constant** for each row of animation there is in your animation sheet. ( I got three rows with different animations ).

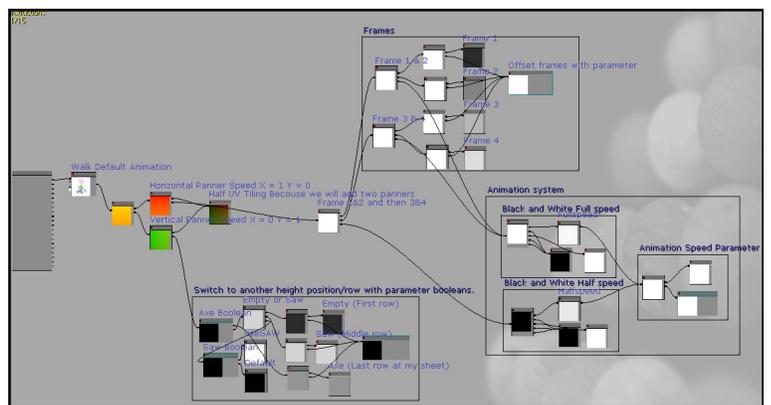
Add one **Add** node for each row as well and a new **Scalar Parameter** like we did for our frames. This can be used to offset the height.



- We can change rows using Switch Parameters.
- Add two **Switch Parameters**.
- Add one **Linear Interpolation(Lerp)** and two **Constants**.

Connect two of the row positions to the Linear Interpolation. Connect a Switch parameter to the Linear Interpolation alpha. Connect both constants to the Switch True and False input. Set one constant to 1 and the other to 0.

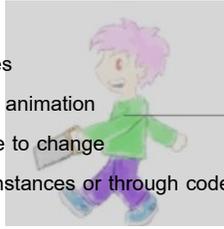
Now we can switch between two height positions.



Connect the Linear Interpolation into the second switch's False input and add the third height position to its True input. Connect this row position system to our second Panner Time. Now its possible to change positions in Y and play new animations that is located on the same animation sheet.

On the next and last page, let's flip the animation horizontally.

We have our animation setup with four frames  
 We can also change rows to choose another animation within the same spritesheet. It's also possible to change the material with custom parameters to our instances or through code.

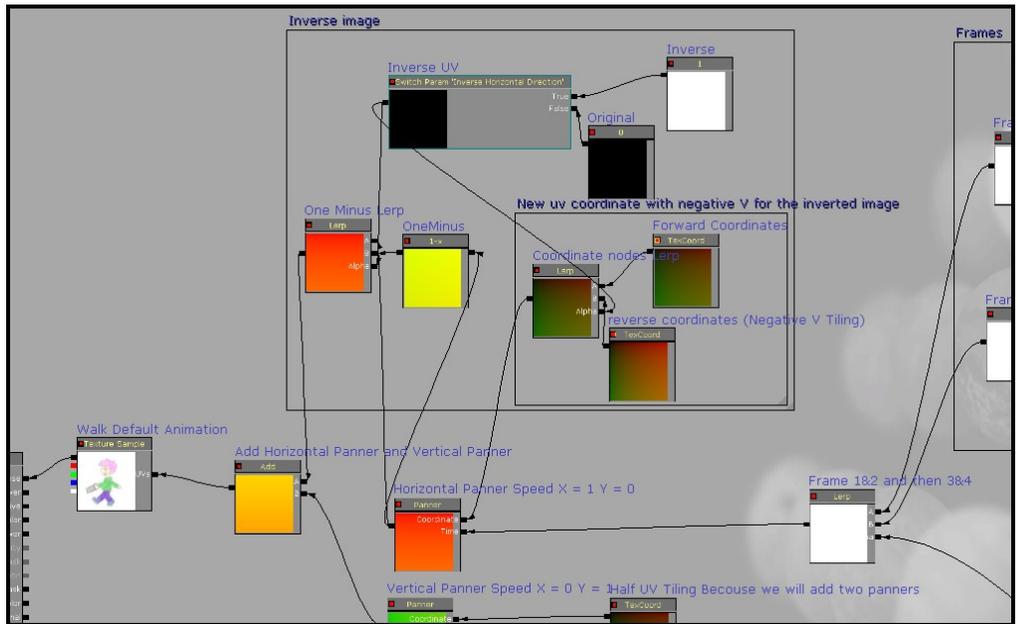
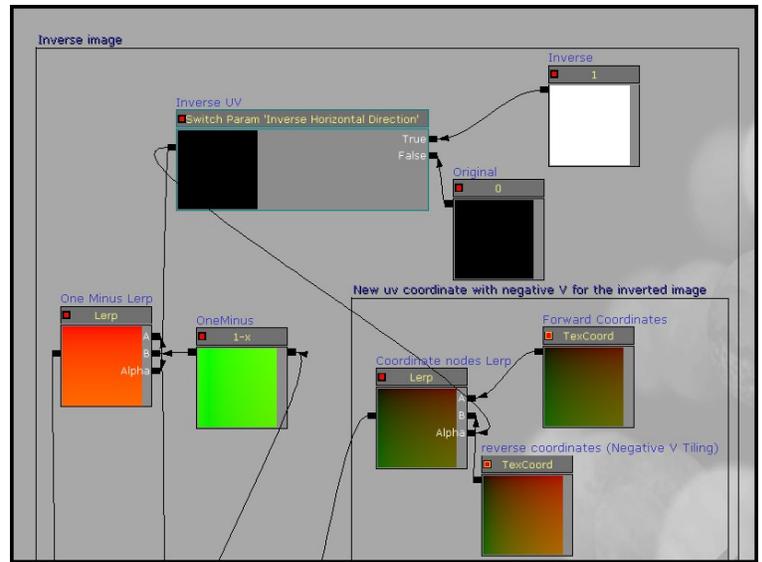


One last thing, we can flip our animation using a OneMinus.



- Add one **Switch**
- Add Two **Linear Interpolations(Lerp)**
- Add two new **Texture Coordinates**.
- Add two **Constants**.
- Add one **OneMinus**

We have to make two Texture Coordinates when one of them is for the reversed direction. Texture Coordinate used for reversed direction may have a value in U that is three time the size of the normal direction nodes value. It will also have a negative version of the normal nodes value in V. This will flip the image at its center, if not, tweak it until its fit later when this function is done.



- Connect the normal (Forward) Texture Coordinate to a Linear Interpolation in A.
- Connect the reversed Texture Coordinate into B and the Switch to it's Alpha.
- Connect both Constants to the Switch True and False, set the constant that goes into True to 1.

- Connect the Horizontal Panner ( The one that drive our frames ) into OneMinus and the Linear Interpolation to its Coordinates.
- Connect the OneMinus to the remaining Linear Interpolation in B and the Horizontal Panner in A.
- Connect Switch to Linear Interpolation Alpha.
- Connect Linear Interpolation to the Add A that's connected to the Texture Parameter.

And we're done!