Peter Vanderhyde

Animation Syntax

Table of Contents

Setting Up the Animation Collection

Properties

- "indexes"
- "<animation_name>"

Indexes

At the beginning of the animation json, there is one property that must be defined; the indexes property. This is a list of names that will be used to refer to each image of the object so that each image can be manipulated by the animation separately. It is order specific, so the order of the names must be in the same order that the images are given to be drawn.

Animation Names

After this property, each animation is created with whatever custom name is desired.

```
example.json

{
    "indexes":["bottom layer","middle layer","top layer"],
    "cool animation":{
        ...
    },
    "death animation":{
        ...
    },
    etc.
}
```

Setting up the animations

Properties

- "duration"
- "initial frame properties"
- "on finish" (optional)
- "loop" (default False)
- "frames" or "create frames" (only if duration is 0)

> Note: The "create frames" property was created for use when Piskel is used so that tens of pictures could easily be made into animations.

Duration

The "duration" property is exactly what is sounds like. It is used to specify the duration of the entire animation in seconds. There is a special case where the duration is set to 0 seconds. This means that the animating component will only use the initial frame given for displaying. In essence, it becomes just a static drawing of the object, but this can be used to reset the appearance of an object if the scene changes or a different animation was in the middle of animating when the animation changed and you need to make sure it looks correct.

On Finish

I will explain the "initial frame properties" in a moment, but first the "on finish" property is an optional list that can be used to specify some action or multiple actions to be taken once the animation has concluded. Some options are more useful than others as some were created for very specific use cases.

Options:

- "destroy component"
- "spawn tank particles"

- "spawn bullet particles"

> - "destroy component" will destroy whatever object the animation component belongs to. This is the most commonly used option as particles, bullets, and most objects have a death or despawn animation that, when it finishes, should logically cause the object to die.
> - "spawn tank particles" will result in the function being called that spawns the particles that spray out from tanks when they die. - "spawn bullet particles" does the same as tank particles but calls the bullet specific function because they have differences.

Loop

This is a very simple property. If not set, the animator will assume that it is false. If it is set to true, then the animation will loop until it is stopped by something externally.

Frame Construction

To talk about the initial frame properties and creating frames, we need to talk about how a frame is constructed.

A frame consists of these parts:

Frame Parts

- "delay"
- "properties"

The delay of a frame is the delay in time from the last frame to this frame. Since the total duration of the animation was defined earlier, this delay is defined as a percentage. It represents what percentage of the total animation duration is spent transitioning from the last frame to this one. If the delay were set as 0.5, if the duration of the animation was 0.2 seconds, that frame would be shown 0.1 seconds after the last frame.

> Delay_____..... 0.5 ..... 0.3 ..... 0.2
> Frames___ |----------->|---------->|------->
> Animation|---------- 0.2 seconds ---------->

The combined percentages add up to 100% and those three frames make up the 0.2 second animation. As long as the delays add up to 100%, the animtion component will handle playing the frames for the correct lengths of time.

The properties of the frame consist of references to each of the images that will be altered that frame.

```
example.json

"properties":{
    "barrel":{
        "scale":1.3,
```

```
            "rotation":26
        },
        "body":{
            "image":"1"
        }
    }
```

The scale and rotation editors are self-explanatory, however with image, it gets a little strange. It checks what string was passed into the animation component as the name for this collection of animations. This must match a key in the animations.py file so that it can reference the correct animation file. It also checks what the name of this current animation was set to. Finally it checks what string is being used to reference this image in the indexes list. It then combines those three elements, uses it as a dictionary key, and retrieves the correct image. This also means that the image must be stored in the correctly named folder structure with the correct name as well.

<collection_name>/<animation_name>/<image_index_name>_<image_key_value>.png

> i.e. player tank/shoot bullet/top layer_1.png

> Note: If the image is changed on the final frame of the animation, it will likely be immediately changed by the next animation or the same animation looping. This means you will likely need to add another frame on the same that doesn't change images so that the final image frame can be viewed for a period of time.

There is actually another property that can be edited. Instead of using one of the strings that refers to an image of the object, the string "sound" can be used to create sounds during the animation.

```
example.json

"properties":{
    "sound":{
        "name":"fire_bullet",
        "volume":0.3
    }
}
```

This will play whatever sound is saved to "fire_bullet" in the SOUNDS list in the settings file at 30% volume.

Phew! That was a lot. Anyway, now that we understand how frames work, we can look at the last couple animation properties. The first

Initial Frame Properties

Phew! That was a lot. Anyway, now that we understand how frames work, we can look at the last couple animation properties. The first is the "initial frame properties" property. This is formatted the exact same as the frame properties that we just discussed. Since there is no delay before the initial frame is drawn, there is no need to use a whole frame; only the properties section to tell us what the object should start out looking like.

> Note: If you don't want your animation to interfere with other possible animations that are also happening, make sure to only initially set and change the pieces of the object that you plan to manipulate in the animation. That means that, if one animation is animating the body of a tank and another is animating the barrel, the two can happen at the same time without affecting each other.
> However, if you want your animation to be the only one occuring, make sure to initially set all of the elements of the object to whatever you want them to be.

Frames

The frames property is also pretty straightforward. It is a list that holds all of the frame objects of the animation in order.

```
example.json

"frames":[
    {
        "delay":0.4,
        "properties":{
            "barrel":{
                "scale":0.6
            },
            "body":{
                "scale":0.6
            }
        }
    },
    {
        "delay":0.1,
        "properties":{
            "barrel":{
                "scale":0.6
            },
            "body":{
                "scale":0.6
            },
            "sound":{
                "name":"pop_low"
            }
        }
    },
    {
        "delay":0.5,
        "properties":{
            "barrel":{
                "scale":1.5
            },
            "body":{
                "scale":1.5
```

```
            }
        }
    }
]
```

Create Frames

if "create frames" is used, there are several things that will be different. The point of this property is so that the animator can generate several frames that all have the same thing changing at the same rate. It can be used to smoothly change the properties of the images. It is set up like so:

```
example.json

"initial frame properties":{
    "effect":{
        "scale":1
    }
},
"create frames":{
    "properties":{
        "effect":{
            "scale":4
        }
    }
}
```

Create frames does not need a delay because each frame will be evenly spaced out across the extent of the duration. In this example, the frames will slowly scale from 1 to 4 over the duration of the animation. The same will work for rotation. However, this is not the primary use of "create frames". This example is missing the key component. Let's look at another example.

```
example.json

"initial frame properties":{
    "effect":{
        "image":"00",
        "scale":1
    }
},
"create frames":{
    "properties":{
        "effect":{
            "image":23,
            "scale":4
        }
```

```
    }
}
```

Here the animation is changing images every frame. Within "create frames", instead of setting "image" to the string suffix of the new image, it is set to the number of images - 1, so the final image to be used would be "effect_23.png". Within the initial properties, "image" is used to determine whether the number of images is under ten, under 100, etc. In the case of the example the two zeros means that there are over 10 images. If there were less, it would only be one zero. The actual numbers in the string don't matter, however the number of ints does.
One final thing to note is that it will create an extra final frame like I mentioned earlier so that the final image frame can actually be seen.