Part 11 - Running and Standing

Our little hero keeps running and facing the same direction.

He needs to be able to stand still when not running. To do this, he needs add an idle animation.

An idle animation would be simply playing hero frame 1 from the spritesheet over and over. Let's create a `HeroIdle` animation in the HeroAnimationSet and set it as the starting animation:

```
[HeroAnimationSet]
Texture = soldier_full.png
FrameSize = (32, 32, 0)
HeroRun = 6
HeroIdle = 1
StartAnim = HeroIdle
```

Like the run animation, we will give the idle animation a slow keyframe speed as it doesn't really need to update often:

```
[HeroIdle]
KeyDuration = 1.0
```

Some links are now needed. Remember, you are starting to build a graph of animations. Now that you have two animations, they can link in several ways:

- Idle to Run
- Run to Idle
- Idle to Idle
- Run to Run

Add all the possible links to `HeroAnimationSet`:

```
[HeroAnimationSet]
Texture = soldier_full.png
FrameSize = (32, 32, 0)
HeroRun = 6
HeroIdle = 1
StartAnim = HeroIdle
HeroIdle-> = HeroIdle # HeroRun
HeroRun-> = HeroRun # HeroIdle
```

The syntax above is a little strange. For example:

```
HeroRun-> = HeroRun # HeroIdle
```

...means that when the HeroRun animation finishes, it could branch off to HeroRun again or HeroIdle, depending what the current target animation is set to.

[KILL THIS PARA] A note on the Priorty = 7 property for the last link. All links have a priority of 8.

When the hero stops running, we want it to calculate back to the HeroIdle animation. We don't know if HeroRunLinkLoop will execute or if HeroRunToIdleLink will. Lowering the priority of HeroRunToIdleLink will ensure our hero will play the idle animation.

When the right key is pressed, the run animation (`HeroRun`) needs to play by setting it as the target animation. And when right is released, the target is set to `HeroIdle`, and the graph must calculate it's way back to “Idle” as the animation to play:

```
if (orxInput_IsActive("GoRight"))
{
    orxObject_ApplyImpulse(hero, &rightSpeed, orxNULL);
    orxObject_SetTargetAnim(hero, "HeroRun");
} else {
    orxObject_SetTargetAnim(hero, "HeroIdle");
}
```

Compile and run. Pressing right will start the run animation, and letting go will revert back to the idle animation. But notice it takes about a second for the animation to start.

This is because of the 1.0 set as the default duration on the [HeroIdle] section. We could reduce this duration value to something smaller, but a more correct way would be to ensure that any request to play the run animation is delivered immediately. Change the HeroIdle link from:

```
HeroIdle-> = HeroIdle # HeroRun
```

to:

```
HeroIdle-> = HeroIdle # .HeroRun
```

The introduction of the `.` symbol in front of the `HeroRun` means to stop executing the `HeroIdle` animation, and to skip immediately to the `HeroRun` animation.

Run that. Much better isn't it?

That takes care of running and stopping while facing right. But what about left?

Let try that now.

Next: Part 12 – Changing Direction.