Realistic Walk Movement

One of the requirements I have for my project is for my character to plant his feet firmly on the ground as he walks. Many games, especially platformers, will have a main sprite with many frames of animation, and will be repositioned across the screen as the character animates.

An interesting thing to notice with many games is that when a character plants his foot on the ground, it glides across without any friction. For most games, this is fine. But where a little more realism is required, some tricks are needed.

Orx has a complete physics system for moving objects around but this tutorial will require using an old school method to move the object.

This tutorial aims to show you how to:

1) Move an object sprite manually.

2) Move the object with a realistic walk motion.

3) Using an EventHandler to update movement in sync with frame changes.

4) Creating custom events in the config, and having the EventHandler use them.

Prerequisite

You are best to be familiar with the Anim Tutorial before this one.

Simple Movement vs a Distance Array

With a simple movement, for each frame change, the object is usually moved x pixels along. With this solution, a list of varying distance values is used. For each frame change, an evant value is taken, and the character is moved that many pixels along.

That is the concept in a nutshell.

I have provided a set of animation frames for this tutorial here.

The graphic is 506 x 100 and each frame varies in width. Save to the data/anim folder.

Define your Config File

```
[DroidAnimationSet]
AnimationList = DroidWalkAnim#DroidIdleAnim
```

```
LinkList    = DroidIdle2Walk#DroidWalk2Idle#DroidIdleLoop#DroidWalkLoop

[DroidIdle2Walk]
Source      = DroidIdleAnim
Destination = DroidWalkAnim
Property     = immediate

[DroidWalk2Idle]
Source      = DroidWalkAnim
Destination = DroidIdleAnim
Property     = immediate
Priority    = 10

[DroidIdleLoop]
Source      = DroidIdleAnim
Destination = DroidIdleAnim

[DroidWalkLoop]
Source      = DroidWalkAnim
Destination = DroidWalkAnim

[DroidIdleAnim]
KeyData1      = droid1
KeyDuration1  = 0.2
Property     = immediate

[DroidWalkAnim]
DefaultKeyDuration  = 0.2;
KeyData1            = droid1
KeyData2            = droid2
KeyData3            = droid3
KeyData4            = droid4
KeyData5            = droid5
KeyData6            = droid6
KeyData7            = droid7
EventName1         = AR1
EventTime1         = 0.2
EventValue1            = 13
EventName2         = AR1
EventTime2         = 0.4
EventValue2            = 6
EventName3         = AR1
EventTime3         = 0.6
EventValue3            = 18
EventName4         = AR1
EventTime4         = 0.8
EventValue4            = 10
EventName5         = AR1
EventTime5         = 1.0
```

```
EventValue5           = 19
EventName6        = AR1
EventTime6        = 1.2
EventValue6           = 14
EventName7        = AR1
EventTime7        = 1.4
EventValue7           = 21
```

The [DroidWalkAnim] section contains the custom events which are very important. Each EventName is set the same as this is not important for this tutorial. But the EventTime and EventValue values are very important, and are defined in order to be in sync with the EventHandler. The KeyDuration for the animation is 0.2 so each EventTime is as well, starting with: 0.2, then 0.4, 0.6, 0.8 etc etc.

We start with 0.2, not 0.0 because we don't want a movement on the first frame.

And each EventValue contains a pixel distance value used to move our character each step. These vary as you can see above: 13, 6, 18, 10, 19, 14 & 21 pixels.

```
[DroidObject]
Graphic           = droid1; This will do as a default.
AnimationSet    = DroidAnimationSet
Position        = (5, 130, 0)

[DroidGraphic]
Texture     = ../data/anim/droid-animation.png
Pivot      = top right

[droid1@DroidGraphic]
TextureCorner = (0, 0, 0)
TextureSize = (63, 100, 0)

[droid2@DroidGraphic]
TextureCorner = (63, 0, 0)
TextureSize = (72, 100, 0)

[droid3@DroidGraphic]
TextureCorner = (135, 0, 0)
TextureSize = (78, 100, 0)

[droid4@DroidGraphic]
TextureCorner = (213, 0, 0)
TextureSize = (78, 100, 0)

[droid5@DroidGraphic]
TextureCorner = (291, 0, 0)
TextureSize = (65, 100, 0)

[droid6@DroidGraphic]
TextureCorner = (356, 0, 0)
TextureSize = (80, 100, 0)
```

```
[droid7@DroidGraphic]
TextureCorner = (436, 0, 0)
TextureSize = (70, 100, 0)
```

You'll notice there is only right hand movement defined, just to keep the tutorial simple.

Setting Up

A few things to set up, first our little droid object:

```
orxOBJECT *droid = orxObject_CreateFromConfig("DroidObject");
orxObject_SetCurrentAnim(droid, orxNULL );
```

Next is to set up an EventHandler for custom animation events

```
orxEvent_AddHandler(orxEVENT_TYPE_ANIM, EventHandler);
```

And the handler:

```
orxSTATUS orxFASTCALL EventHandler(const orxEVENT *_pstEvent){
    orxANIM_EVENT_PAYLOAD *pstPayload;
    pstPayload = (orxANIM_EVENT_PAYLOAD *)_pstEvent->pstPayload;

    switch(_pstEvent->eID){

        case orxANIM_EVENT_CUSTOM_EVENT: {

            orxLOG("<%s>@<%s> was fired PIXELS %f ",  pstPayload->zAnimName,
orxObject_GetName(orxOBJECT(_pstEvent->hRecipient)),
pstPayload->fCustomEventValue );

            orxVECTOR droidVector;
            orxObject_GetPosition(droid, &droidVector);
            droidVector.fX = droidVector.fX + pstPayload->fCustomEventValue;
            orxObject_SetPosition(droid, &droidVector);

            break;
        }

    }

    return orxSTATUS_SUCCESS;
}
```

In the event handler, the orxANIM_EVENT_CUSTOM_EVENT is fired because the EventTime(s) having been defined in the config. The current position is added to by retrieving the active EventValue (pstPayload→fCustomEventValue) from the config file.

Setting the proper animation with key on and key off is done with:

```
    if(orxInput_IsActive("Right") && orxInput_HasNewStatus("Right")){
        orxObject_SetTargetAnim(droid, "DroidWalkAnim" );
    }

    if(!orxInput_IsActive("Right") ){
        orxObject_SetCurrentAnim(droid, "DroidIdleAnim"); //currently used
instead of orxObject_SetTargetAnim(droid, orxNULL );  due to a small bug in
orx 1.2 (fixed in SVN)
    }
```

When you run the program, the droid's feet should plant nice and firmly on the ground as it walks along.

Possible pitfalls:

1) Be careful to make all your Config EventTime(s) different. Two EventTime(s) the same will crash.