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

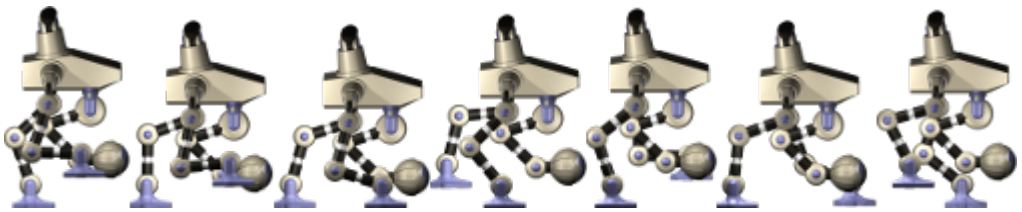
You are best to be familiar with the Anim Tutorial before this one.

Fixed Distance vs Varying Distance

With a simple movement, for each frame change, the object is usually moved a certain amount of pixels along. With this solution, a list of varying distance values is used. For each frame change, an event 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

```
[DroidObject]
Graphic        = DroidGraphic ; Default frame size for all the animation
frames.
AnimationSet   = DroidAnimationSet
Position   = (5, 130, 0)

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

[DroidAnimationSet]
Texture       = droid-animation.png
KeyDuration   = 0.2
Digits        = 2
StartAnim     = DroidIdleAnim
DroidIdleAnim = 1
DroidWalkAnim = 7
DroidIdleAnim-> = DroidIdleAnim # .DroidWalkAnim
DroidWalkAnim-> = DroidWalkAnim # .DroidIdleAnim

[DroidIdleAnim]
Direction = left # up ;get frames from right to left, in this case: get a
single frame from the right

[DroidIdleAnim01]
TextureSize = (70, 100, 0)
KeyEvent    = AR1 # 0 ;move 0 distance when idle

[DroidWalkAnim01]
TextureSize = (63, 100, 0)
KeyEvent    = AR1 # 30 ;move 30 pixels right on this frame

[DroidWalkAnim02]
TextureSize = (72, 100, 0)
KeyEvent    = AR1 # 5 ;move 4 pixels right on this frame, etc

[DroidWalkAnim03]
TextureSize = (78, 100, 0)
KeyEvent    = AR1 # 0

[DroidWalkAnim04]
TextureSize = (78, 100, 0)
KeyEvent    = AR1 # 19

[DroidWalkAnim05]
TextureSize = (65, 100, 0)
```

```
KeyEvent    = AR1 # 22

[DroidWalkAnim06]
TextureSize = (80, 100, 0)
KeyEvent    = AR1 # 4

[DroidWalkAnim07]
TextureSize = (70, 100, 0)
KeyEvent    = AR1 # 26
```

The `[DroidWalkAnimXX]` sections contains the individual frame sizes. Normally we use a `FrameSize` in the AnimSet, but because our frames are all different sizes we specify them manually.

Also in the `[DroidWalkAnimXX]` are the custom key events which are very important. The first value is the event name and the second value is the event value.

Each event value contains the pixel distance value used to move our character each step. These vary as you can see above: 30, 5, 0, 19, 22, 4 & 26 pixels.

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 in the Init() function:

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

And then the object variable at the top of the code:

```
#include "orx.h"
orxOBJECT *droid;
```

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->stCustom.fValue );

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

            break;
        }

    }

    return orxSTATUS_SUCCESS;
}
```

In the event handler, the orxANIM_EVENT_CUSTOM_EVENT is fired because the KeyEvents having been defined in the config. The current position is added to by retrieving the active KeyEventValue (pstPayload→stCustom.fValue) from the config file.

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

```
  if (orxInput_IsActive("Right")) {
      orxObject_SetTargetAnim(droid, "DroidWalkAnim");
  }
  else {
      orxObject_SetTargetAnim(droid, "DroidIdleAnim");
  }
```

And you'll need a key map:

```
[MainInput]
KEY_ESCAPE = Quit
KEY_RIGHT  = Right
```

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