

Obsidian API Tutorial #4 – Animations – All versions

<< Previous (<http://www.dabigjoe.com/obsidian-suite/tutorials/3a-entity-shadow-size-scale/>) All tutorials (<http://www.dabigjoe.com/obsidian-suite/tutorials/>) Next >> (<http://www.dabigjoe.com/obsidian-suite/tutorials/5-entity-ai>)

Let's add some animations

Okay, this is the tutorial you've all been waiting for, the one where we actually get some animations working! The video for this tutorial can be found here (https://youtu.be/g95_qUgqnik).

Prerequisites

You'll need an entity setup as per the previous tutorial, and some animations ready to go!

Registering an entity

So in the last tutorial, we setup an entity so that it is considered 'animated' – that is it can make use of the API functionality and hence use Obsidian Animations. The first step in animating it in game (after creating the animations using The Animator of course), is to tell the AnimationRegistry about our entity. This is done in the CommonProxy, in the registerAnimations method we created in the second tutorial (<http://www.dabigjoe.com/obsidian-suite/tutorials/2-resources/>). The line we need is:

```
AnimationRegistry.registerEntity(EntitySaiga.class, "saiga");
```

The first argument is the entity class you want to register, and the second argument is the name you want to register your entity with.

Registering Animations

This is the trickiest part of this tutorial. We use another AnimationRegistry method called registerAnimation which takes *six* arguments, and it can seem pretty daunting at first, but don't worry, I'll take you through it step by step.

First of, you need to make sure you get the correct method, as there are several different ones. The one you want is:

```
registerAnimation(String (http://www.google.com/search?hl=en&q=allinurl%3Adocs.oracle.com+javase+docs+api+string) entityType, String
```

Let's break it down...

The first three

The first three arguments are straightforward. We want to register an animation to a specific entity, so we need to tell the AnimationRegistry which entity we're registering for. This is done with the first argument, entityType. In my example, it will be "saiga". This **must** match the name you registered the entity class with earlier.

The second argument, the binding, is a name given to the animation we're registering. In my case, I'm registering a walk animation, so I'll call mine "walk".

The third argument is a ResourceLocation, and this is the resource that we sorted out in the second tutorial, (<http://www.dabigjoe.com/obsidian-suite/tutorials/2-resources/>) in my case saigaWalk. Okay so far?

Animation Priority

The fourth argument is the priority of the animation. Higher priority animations will run in preference to lower priority animations if both animations are 'active'. So, for example, if a player is running but also swinging their sword, you want the sword swing animation to play rather than just the running animation (at least for the top half of the body – more on that later). Therefore, the sword swing animation would have higher priority than the running animation. **A higher priority means a lower value**, e.g. 1 is higher priority than 2. This is a programming convention.

So in my case, I've added two animations – walking and idle (not walking). Walking has higher priority than idling, so a value of 0 for walking and 50 for idling.

Looping

The fifth argument is a Boolean value that signifies whether an animation should loop, i.e. start over once it has finished. Both walking and idling are examples of looping animations, you want the animation to continue until the player does something else. An example of a non looping animation would be jumping (assuming the player only jumps once).

IsActiveFunction

This is a Java Lambda Function, I suggest you go read up on them if you're not sure what they are. The argument expects an IsActiveFunction, a custom type of Lambda function I created, which determines if the animation is 'active'. By active, I mean the entity is doing something that should trigger the animation, e.g. if they're walking, the walk animation should play; if they're eating, the eat animation should play.

Let's have a look at the IsActiveFunction for walking:

```
IsActiveFunction isWalking = (entity) -> {
    return ObsidianAPIUtil.isEntityMoving(entity);
};
```

So, we define the isWalking function as follows. It expects one argument, an entity, and then it returns a boolean by doing some processing on this entity, denoted by (entity) -> {return someBool}. In this example, I use the ObsidianAPIUtil class to check if the entity is moving using the isEntityMoving function. In more advanced cases, you would do more processing to check if an entity is jumping, swinging a pick,

biting etc., but hopefully this is illustrative enough to get the functionality across.

A little on how it works

If that all seems a little complex, I suggest watching this section (https://youtu.be/g95_qUgqnik?t=4m8s) of my video for this tutorial – it goes into a bit more depth. I'll also cover a bit of what happens in the background, which should help clarify things.

Every render tick, when the game is rendering an entity, the renderer starts going through the list of animations registered for an entity, from highest priority to lowest priority. First, the renderer checks the highest priority animation – if its `isActiveFunction` returns true, then the renderer uses that animation to draw the entity. If its `isActiveFunction` returns false, the renderer moves on to the next highest priority animation and repeats. It does this until it finds an active animation, or until it runs out of animations to check, in which case the entity is rendered in its default state, i.e. how it looks when modelled.

Putting it all together

Hopefully, you now understand how to register animations. Below are the final lines of code I use for registering the Saiga walk animation:

```
IsActiveFunction isWalking = (entity) -> {
    return ObsidianAPIUtil.isEntityMoving(entity);
};
```

```
AnimationRegistry.registerAnimation("saiga", "walk", saigaWalk, 0, true, isWalking);
```

That will register the walk animation, which will now play that animation in game! Run Minecraft and give it a go.

A word on idle animations

I said earlier that the entity will be rendered in its static state if no animations are active, which looks terrible. An easy way to bring your entities to life is by giving them an idle animation – a simple head bob and tail wag for example. Idle animations are the lowest priority animations; they should only happen when no other animations are active. You also always want them to be considered active, so the entity is never rendered statically. We can use a special `IsActiveFunction` for this:

```
IsActiveFunction returnTrue = (entity) -> {
    return true;
};
```

It always returns true! This guarantees this animation will always be rendered if the renderer reaches it when it's deciding which animation it should use, so it must be the lowest priority (otherwise animations with lower priority will never, ever get selected). In my example, I registered the Saiga idle animation like so:

```
AnimationRegistry.registerAnimation("saiga", "idle", saigaIdle, 50, true, returnTrue);
```

All the other Saiga animations must have a priority greater than this (i.e. priority < 50).

Right, that's all the basics!

Congratulations, you made it! Everything beyond this point is extra – you have (pretty much) everything you need to start using all your animations now. The next tutorial will focus on tying animations to EntityAI.

<< Previous (<http://www.dabigjoe.com/obsidian-suite/tutorials/3a-entity-shadow-size-scale/>) All tutorials (<http://www.dabigjoe.com/obsidian-suite/tutorials/>) Next >> (<http://www.dabigjoe.com/obsidian-suite/tutorials/5-entity-ai>)

Edited files

CommonProxy

```
public class CommonProxy {

    private ResourceLocation saigaWalk = new ResourceLocation("mod_api_tutorial:animations/Walk.oba");
    private ResourceLocation saigaIdle = new ResourceLocation("mod_api_tutorial:animations/SaigaIdle.oba");

    public void init()
    {
        ModEntities.registerEntities();
        registerRendering();
        registerAnimations();
    }

    public void registerRendering() {}

    public void registerAnimations() {
        AnimationRegistry.init();


        IsActiveFunction isWalking = (entity) -> {
            return ObsidianAPIUtil.isEntityMoving(entity);
        };

        IsActiveFunction returnTrue = (entity) -> {
            return true;
        };
    }
}
```

```
AnimationRegistry.registerEntity(EntitySaiga.class, "saiga");
AnimationRegistry.registerAnimation("saiga", "walk", saigaWalk, 0, true, isWalking);
AnimationRegistry.registerAnimation("saiga", "idle", saigaIdle, 50, true, returnTrue);
}
}
```

1 Download Template (Free) To View Template, Download Here [quicktemplatefinder.com](#) 

2 You will love Freshdesk. Web-based Helpdesk Software for Service Desks & Support Centers. [freshdesk.com](#) 

 Edit (<http://www.dabigjoe.com/wp-admin/post.php?post=539&action=edit>)

LEAVE A REPLY

Logged in as dabigjoe (<http://www.dabigjoe.com/wp-admin/profile.php>). Log out? (http://www.dabigjoe.com/wp-login.php?action=logout&redirect_to=http%3A%2F%2Fwww.dabigjoe.com%2Fobsidian-suite%2Ftutorials%2F4-animations%2F&_wpnonce=6e8ffaa668)

Comment

POST COMMENT