

Uberlisk

SCZMAPSTER.COM

By

Hati(xhatix)

Uberlisk?

- Simply, Blizzard presented the Uberlisk at BlizzCon09
 - They said it was done by one of their skilled employees within ~1.5h
 - The Uberlisk has SpineCrawlers attached to his head which are attacking his foes, he can do seismic eruption in front of him changing the terrain, and killing nearly foes if he dies.



Read before Next step

I will **not** fully explain how I added 5 Spinecrawlers to a Ultralisk

I will just give you a slight overview about all this, as explaining everything would make me to write ~1day for just this which isn't my intention really today ,

I will explain how to add one(1) Spinecrawler to the ultralisk, the following Spinecrawlers are simply copy&paste/duplications of the others

I might miss something, if so post in forum/irc

If you prefer going through the dirt – just go on with understanding the files in the zip
(uberliskfiles.zip)

Let's get started.

Creating Uberlisk

- First you need, if you still want to have a Ultralisk, to create a new Unit, preferably you make a copy of Ultralisk and 'id' it Uberlisk
UnitData.xml:



```
<CUnit id="Uberlisk">  
    <Name value="Uberlisk"/>  
    ...Copy of Ultralisk tags/Attributes...  
</CUnit>
```

Let's get started.

Creating Uberlisk

- Next step is creating the Actor, which handles: animations, effects, model, sound, portrait and many other things
- Again the easiest/fastest way to go on with this step is copy&pasting the Ultralisk entry, ,id'-ing it Uberlisk



```
<CActorUnit id="Uberlisk" parent="GenericUnitBase">  
    ...Tags/Attributes...  
</CActorUnit>
```

Let's get started.

Creating Uberlisk

- To have appropriate animations, and sounds you should insert the following

```
<SoundArray index="Birth" value="Ultralisk_Birth"/>
<SoundArray index="Ready" value="Ultralisk_Ready"/>
<SoundArray index="Help" value="Ultralisk_Help"/>
<SoundArray index="What" value="Ultralisk_What"/>
<SoundArray index="Yes" value="Ultralisk_Yes"/>
<SoundArray index="Attack" value="Ultralisk_Attack"/>
<SoundArray index="Pissed" value="Ultralisk_Pissed"/>
<SoundArray index="Movement" value="Ultralisk_Movement"/>
<SoundArray index="Turning" value="Ultralisk_Turning"/>
<SoundArray index="Board" value="Ultralisk_Board"/>
<SoundArray index="Click" value="Ultralisk_Click"/>
<SoundArray index="ClickError" value="Ultralisk_ClickError"/>
<SoundArray index="Highlight" value="Ultralisk_Highlight"/>
<GroupSoundArray index="What" value="Ultralisk_GroupWhat"/>
<GroupSoundArray index="Yes" value="Ultralisk_GroupYes"/>
<GroupSoundArray index="Attack" value="Ultralisk_GroupAttack"/>
<GroupSoundArray index="Board" value="Ultralisk_GroupBoard"/>
<HighlightTooltip value="Unit/Name/Ultralisk"/>
<On Terms="UnitBirth.Uberlisk" Send="Create SoundEmitter Ultralisk_Ambience"/>
<On Terms="UnitBirth.Uberlisk" Send="Create"/>
<On Terms="UnitRevive.Uberlisk" Send="Create"/>
```

This is playing the ,common' (Ultralisk's) Sounds if you create,kill, etc. the Uberlisk

Let's get started.

Creating Uberlisk

- If you would start the game right now, with a placed Uberlisk, you should see the Test-Sphere, this is because we currently have no model called “Uberlisk” to create it go into ModelData.xml and add the following

```
<CModel id="Uberlisk" parent="Unit" Race="Zerg">
    <Model value="Assets\Units\##Race##\Ultralisk\Ultralisk_00.m3"/>
    <Occlusion value="Show"/>
    <ScaleMax value="1.0000,1.0000,1.0000"/>
    <ScaleMin value="1.0000,1.0000,1.0000"/>
    <SelectionRadius value="1.1762"/>
    <Radius value="1.250000"/>
</CModel>
```

We are using another Model than normal Ultralisk does, as this can attach SpineCrawlers
(+ copy the ,AttachProps' from Ultralisk in it too (not in here because of space)

Or just go here: [pastebin](#)

Let's get started.

Give the Uberlisk a new weapon

- So next step to do is – giving the Uberlisk a weapon

To give a unit an Weapon you require the following things:

- WeaponArray in UnitData.xml
 - ie. <WeaponArray Link="UltraliskRange" />
- a valid entry in WeaponData.xml
- the correspondenting effects in EffectData.xml
- and as you are we are using a missile, we also require a Mover in MoverData.xml
- a missile-unit

Let's get started.

Give the Uberlisk a new weapon

UnitData.xml

To add a new weapon to your Unit you have to go into the UnitData.xml and add a new attribute/tag to your unit

```
<CUnit id="UnitID">
    <...Stuff.../>
    <WeaponArray id="WeaponData.xml-ID"/>
    <...otherStuff.../>
</Cunit>
```

Let's get started.

Give the Uberlisk a new weapon

WeaponData.xml

My process to add a new Weapon starts with placing this default template in it

```
<CWeaponLegacy id=„id“>
    <Options index="Hidden" value="0"/>
    <DisplayEffect value="DisplayEffect"/>
    <TargetFilters value="Valid;NotValid"/>
    <Range value="Range"/>
    <Period value="Atkspd"/>
    <MinScanRange value="scansrange"/>
    <DamagePoint value="animationdelay"/>
    <Arc value="facing"/>
    <Effect value="WeaponEffect"/>
    <!--AllowedMovement value="Moving"-->
    <Backswing value="BlockOtherWeaponForXs"/>
    <Icon value="icon"/>
    <Options index="OnlyFireWhileInAttackOrder" value="1"/>
    <Options index="OnlyFireAtAttackOrderTarget" value="1"/>
    <Options index="LinkedCooldown" value="1"/>
    <Options index="DisplayCooldown" value="1"/>
    <LegacyOptions index="KeepChanneling" value="0"/>
    <LegacyOptions index="CanRetargetWhileChanneling" value="0"/>
    <LegacyOptions index="NoDeceleration" value="0"/>
    <Options index="ContinuousScan" value="1"/>
</CWeaponLegacy>
```

This has mostly any necessary thing in it

After I placed it into the file I just change the stuff to my needs. You will see in next steps

Let's get started.

Give the Uberlisk a new weapon

EffectData.xml

As we are up to create a missile our first/main-effect will be a LaunchMissile one – this allows us to create a Unit(Missile) that follows a Mover (MoverData.xml)

```
<CEffectLaunchMissile id="UltraliskRangeLM2">
    <EditorCategories value="Race:Terran"/>
    <ImpactEffect value="UltraliskRangeU"/>
    <AmmoUnit value="UltraliskRangeLMWeapon"/>
    <Flags index="Return" value="1"/>
    <Movers Link="UltraliskRangeWeapon" />
    <ReturnMovers Link="UltraliskRangeWeaponRetract"/>
    <ReturnDelay value="0.175"/>
</CEffectLaunchMissile>
```

This Effect does the following: If the Missile we launched impacts the target (can be a point too) the ImpactEffect will be fired, the AmmoUnit is required for a later step, to make the SpineCrawler act alive, the ReturnFlag enables things like Boomerang – so the Missile will return to LaunchLocation (default: SourceUnit) after ImpactEffect, the Movers is the Mover used for the missile on the way **towards** the target, while the ReturnMovers is used for it's return to the LaunchLocation, the ReturnDelay is measured in 1s (so it's 0.175s) and allows the missile to 'stick' at a target for a duration, ie. launching a shock-trap to a location giving an AoE effect on it, and you want the shock-trap to be placed there for a while. (the AoE-Duration)

Let's get started.

Give the Uberlisk a new weapon

EffectData.xml

A Impact Effect can be anything (I'm not quite sure here, maybe another missile can be launched too, to create chain lightning)

But we will stay with

```
<CEffectDamage id="UltraliskRangeU" parent="DU_WEAP">
    <EditorCategories value="Race:Terran"/>
    <Kind value="Ranged"/>
    <Amount value="100"/>
    <Death value="Blast"/>
</CEffectDamage>
```

This is a simple DamageEffect, for Ranged weapons, causing 100hitpoints damaged, parented to a default DU_WEAP having other things in it, and has the Death-Type ,Blast' – which mostly let target's explode if they are killed by **this** effect

Let's get started.

Give the Uberlisk a new weapon

MoverData.xml

A Mover is simply the following: a force to guide the missile to it's target

```
<CMoverMissile id="UltraliskRangeWeapon">
    <MotionPhases>
        <Driver value="Throw"/>
        <Acceleration value="50"/>
        <Speed value="50"/>
        <MaxSpeed value="200"/>
        <IgnoresTerrain value="1"/>
        <Outro value="0,2,0,4"/>
        <RotationLaunchActorType value="Supplied"/>
        <ThrowRotationType value="LookAtTarget"/>
        <ThrowVector value="0,-1,6"/>
    </MotionPhases>
    <MotionPhases>
        <Driver value="Guidance"/>
        <Acceleration value="140"/>
        <MaxSpeed value="200"/>
        <IgnoresTerrain value="1"/>
        <TurnType value="RevertToUp"/>
        <YawPitchRoll value="180"/>
        <YawPitchRollAccel value="1440"/>
    </MotionPhases>
    <MotionPhases>
        <Driver value="Guidance"/>
        <Acceleration value="140"/>
        <Speed value="12"/>
        <MaxSpeed value="200"/>
        <IgnoresTerrain value="1"/>
        <TurnType value="RevertToUp"/>
        <RotationLaunchType value="LookAtTarget2D"/>
        <RotationLaunchActorType value="Supplied"/>
        <YawPitchRoll value="MAX"/>
    </MotionPhases>
</CMoverMissile>
```

As you see, there are different MotionPhases in one Mover, each of them played after Outro is 'over' the one showed is for the Move to the target

Let's get started.

Give the Uberlisk a new weapon

UnitData.xml

To create a Missile unit you simply create a new unit in UnitData.xml and parent it to „MISSILE“ or „MISSILE_INVULNERABLE“

```
<CUnit id="UltraliskRangeLMWeapon" parent="MISSILE">
    <Race value="Terr"/>
    <Mover value="UltraliskRangeWeapon"/>
    <EditorCategories value="ObjectType:Projectile, ObjectFamily:Melee"/>
</CUnit>
```

The Mover-attribute is **not** necessary at this point, as you define your own in EffectData.xml if you would not define a Mover in it – it would take the one define in UnitData.xml – **but i dunno how it handles Return with it**

Intermediate

Check if the uberlisk attacks

Start your Starcraft2 – look if your Weapon deals damage, and check uberliskfiles if errors occur – or ask what might help ya out (mostly at bottom of each file)

Missing parts till here: (iirc)

Return Mover – so it might just attack once

ActorData.xml

- The “most important” part – if you want the Uberlisk to act “real” and let the SpineCrawlers attack
- What we are going to do is basically – if the new weapon we gave the Uberlisk is fired the SpineCrawler we are going to attach to the Uberlisk’s Head to follow the missile, to the target, follow it right back to the Uberlisk and go on
- So that’s why I don’t explain that much before, because all this can easily be understood and change while ActorData.xml is might one of the most powerful but also most complex ones (I guess GE delays because of just this one data)
- Some general information for ActorData.xml
 - You can change the unit model on the fly (marine shield)
 - There are logic operations in this files
 - You can do many fancy things just with this data

ActorData.xml

First of all we are going to add the SpineCrawler to the Uberlisk, without giving it animations

Well that's quite easy. Basically all you do is adding a **model** to the Uberlisk.
Code:

Explanation:

We basically scale first down the Model to 60% of it's default size, to make the attachment look better, next we select the Model (SpineCrawler here) to be

used, this way we don't have to change ModelData.xml for every SpineCrawler, next we select the Host, this defines to what unit we want to attach it, later specified by <On>.

HostSiteOps defines **where** on the Model the attachment point is. We changed the Uberlisk model to Ultralisk_00.m3 earlier – to make this possible, as Uberlisk.m3 has **no** attachment points as it seemed

The <On/>'s are like If-cases, so if Uberlisk is born we create this Model, if the Model (Actor) is created we apply the Burrow animation (so it looks like it is burrowed & able to attack)

The WeaponStart Terms, are required to enable the SpineCrawler "react" on the weapon

The Signal is therefore used if the SpineCrawlerTentacle is returning (ReturnMovers)

```
<CActorModel id="UberliskSpineCrawlerTentacle"
parent="ModelAddition">
    <Scale value="0.6"/>
    <Model value="SpineCrawler"/>
    <Host Subject="_Selectable"/>
    <HostSiteOps Ops="SOpAttachWeapon01"/>
    <HostedAttaches Name="TentacleAttack">
        <AttachQuery Methods="Weapon"/>
    </HostedAttaches>
    <On Terms="UnitBirth.Uberlisk" Send="Create"/>
    <On Terms="ActorCreation" Send="AnimGroupApply
Burrow"/>
    <!-- Custom attack anims.
        -->
    <On Terms="WeaponStart.UltraliskRange.AttackStart;
AnimPlaying Burrow" Send="AnimClear Burrow"/>
    <On Terms="WeaponStart.UltraliskRange.AttackStart"
Send="AnimPlay Attack Attack PlayForever"/>
    <On Terms="WeaponStop.UltraliskRange.AttackStop"
Send="AnimClear Attack"/>
    <On Terms="Signal.*.TentacleRetracted" Send="AnimClear
Attack"/>
    <On Terms="Signal.*.TentacleRetracted" Send="RefClear
TentacleAttack"/>
    <On Terms="Signal.*.TentacleRetracted" Send="AnimPlay
AttackEnd Attack,End"/>
</CActorModel>
```

ActorData.xml

This Actor is therefor used for the Missile we are using to control the SpineCrawler, the parent GenericTentacleMissile just includes the <IsTentacle/>-Attribute & MissileTentacleReturn on the ReturnMovers too (so the missile don't stuck at the target), the UnitName defines the missile we are using

Model value="Invisible" – gives the missile an Invisible model so we don't require to give it one in ModelData, and we are not 'wanting' to be there any, the outcommented line is just for debugging reasons

Supporter includes all datas related to Launch

HostReturn includes additional files required for ReturnMove

If the Missile is created we set a reference to

UltraliskRangeAttackMissilesScope, allowing us to link the SpineCrawler spike to link to the missile (following it)

MotionPhaseStart is fired if a MotionPhase in a Mover starts, here it is the first appearance (; MotionPhase, there could also be MotionPhase 2 iirc)

```
<CActorMissile id="UltraliskRangeAttackMissile" parent="GenericTentacleMissile"  
unitName="UltraliskRangeLMWeapon">  
<!--On Terms="MotionPhaseStart" Send="Create IndicatorMissilePhaseChange"/-->  
<Model value="Invisible"/>  
<Supporter>  
<Subject value="UberliskSpineCrawlerTentacle"/>  
<Scope value="Caster"/>  
<Actor value="Find"/>  
<Effect value="UltraliskRangeLM"/>  
</Supporter>  
<HostReturn>  
<Subject value="UberliskSpineCrawlerTentacle"/>  
<Scope value="Target"/>  
<Actor value="Find"/>  
<Effect value="UltraliskRangeLM"/>  
</HostReturn>  
<Remove Terms="UnitBirth" Send="AnimBracketStart Lifetime Birth Stand"/>  
<On Terms="UnitBirth" Send="AnimPlay Main Birth"/>  
<On Terms="ActorCreation" Send="RefSet ::scope.UltraliskRangeAttackMissileScope  
::Self"/>  
<On Terms="SupporterDestruction" Send="$Death"/>  
<On Terms="Effect.*.Return" Send="AnimPlay Main Back PlayForever"/>  
<On Terms="MotionPhaseStart; MotionPhase" Target="::Supporter" Send="RefSet  
TentacleAttack ::Self"/>  
<On Terms="ActorDestruction" Target="::Supporter" Send="Signal  
TentacleRetracted"/>  
</CActorMissile>
```

ActorData.xml

The ActorSite creates a Link to a specific Host (TentacleSpike in this case, define by RefSet) – but not sure here, it works so this is only a thing to get out in time ... Or wait till GE to understand

CActorAction handles the Missile itself

The LaunchSite was defined before, we attach it Over the head of the unit to make the tentacle attack look quite real (AttachQuery & SiteOps), makes the Missile fire from top of the unit

The ImpactAttachQuery & ImpactSiteOps define where on the target the TentacleSpike should be placed/attached

The DamageAttachQuery defines where the damageeffect should be displayed(?)

<Missile/> define what missile should be followed

ImpactMap is being used for the different types of “damage” (Blast, None, Fire etc.) and which effect/sound should be displayed/played at this point

```
<CActorSite id="UberliskSpineCrawlerTentacleLaunchSite">
<Host Subject="::scope.Tentacle"/>
</CActorSite>

-----<CActorAction id="UberliskSpineAttack" parent="GenericAttack"
effectImpact="UltraliskRangeU" effectLaunch="UltraliskRangeLM">
<LaunchSite value="UberliskSpineCrawlerTentacleLaunchSite"/>
<LaunchAttachQuery Methods="Overhead"/>
<LaunchSiteOps Ops="SOpAttachOverhead"/>
<ImpactAttachQuery Methods="Target"/>
<ImpactSiteOps Ops="SOpAttachHarness SOpForwardLaunchGuide"/>
<DamageAttachQuery Methods="Overhead"/>
<Missile value="UltraliskRangeAttackMissile"/>
<ImpactMap index="None" Model="SpineCrawlerAttackImpact"
ModelReaction="SpineCrawlerAttackImpactReaction"
Sound="SpineCrawler_AttackImpact"/>
<ImpactMap index="LightArmor" ModelReaction="LightArmorTargetImpact"/>
<ImpactMap index="Metal" ModelReaction="MetalTargetImpact"/>
<LaunchAssets Model="SpineCrawlerAttackLaunch" Sound=""/>
</CActorAction>
```

Conclusion

I recommend to still watch into the files to understand this fully otherwise you might not get it.

Sry for any failures in grammar/orthographic ...
wrote it while watching tv ...