

# Olander's Realistic Weather for Unity

Tuesday, March 10, 2015  
5:47 PM

**Olander's Realistic Weather** is a feature rich realistic weather system for the incredible Time of Day Lighting for Unity. This weather system add-on is a 'port' from Olander's Realistic Weather (ORW) that was created for the Neverwinter Nights 1 Aurora Engine (also NWN2). This is being provided as a Community Weather System for Free for the very best in Weather for Unity. With minor adjustments this will work for Non-Time of Day lighting systems as well.  
\*The Free does not mean a crap add-on. :)

## Features Included:

- Temperature Climates (6) (Customizable for more...Excel Spreadsheet included to calculate monthly ranges)
  - 1 - Tropical 12 Months Warm 20 to 50
  - 2 - Subtropical 8 Months Warm 20 to 35, 4 at 15 to 25
  - 3 - Temperate 8 Months at 15 to 30, 4 at 2 to -10
  - 4 - Cool 8 Months at 10 to 20, 4 at 10 to -5
  - 5 - Cold 4 Months 5 to 10, 8 at 5 to -30
  - 6 - Polar 12 Months -5 to -50
- Precipitation Climates (5)
  - 0 - Arid - No Rain (even if Humidity is set)
  - 1 - Semi Arid - Slight Summer Rain
  - 2 - Arid Oceanic - Slight Winter Rain
  - 3 - Oceanic - Winter Rain, Dry Summer
  - 4 - Tropical - Summer Rain, Dry Winter
  - 5 - Temperate - Rain in all Seasons
- Water Content of Climate (affects humidity accumulation)
- Humidity %
- Temperature (C & F)
- Wind
- Cold and Warm Fronts. Warm fronts accumulate moisture and temperature while Cold Fronts trigger precipitation.
- Rain and Snow that look very realistic and uses the Shuriken Particle System exclusively. No Instantiations.
- Rain and Snow that varies depending upon the amount of humidity (moisture) that is in the air.
- Rain and Snow at the same time when the temperature is between 0-1.5 C
- Rain Loops (1) (High Quality)
- Thunder (9) (High Quality)
- Lightning that is beautiful to behold
- Lightning Flashes (ambient lighting effect)
- Dynamic Cloud Density based upon humidity
- Darkening and Lightening Dynamic Clouds
- Darkening and Lightening Sun and Moon light depending upon cloud density

All highly tested to work perfectly with Time of Day.

## Included in the Package:

- SkyDome configuration Pics (see the end of this PDF for my setup pics).
- ORW Temperature Zone Calculator (MSEXcel)

Audio WAV (Very High Quality):

- (1) Rain (Stereo but set to Mono)
- (9) Thunder (Mono)

Effects (All Shuriken Particle Effects):

- (3) Lightning (2k textures)
- (1) Rain Drop
- (1) Rain Splash
- (1) Rain Ripple
- (1) Snow Flake
- (1) Snow Ice (denser flake)

\*\*Prefabs for each of the above included

## Game Objects and Scripts:

- \_GameData Prefab (with scripts attached and set up)
- \_WeatherMaster Prefab (with scripts attached and set up)
- \_WeatherObject Prefab (with scripts attached and set up)
- Olander's Realistic Weather Namespace to encapsulate Database Functions, and Weather Functions
- Time script (.cs)
- Olander's Realistic Weather scripts (.cs)

## Persistent Time and Weather:

- Weather is fine as static but really shines when there are a calendar, seasons, and daily sun cycle.
- Time is Database stored (every 30 seconds)
- Time has a configurable Month, Week, and Day calendar (Example Grey Elven and English have been provided)
- Time has a Minutes Per Day setting to compress or stretch days smoothly. This converts the Frame Time into a new Ratio Frame Time that is based upon how many seconds are in One Day. The new Frame Time is accumulated in Seconds just as Time.deltaTime would be each frame
- Please see gTime.cs for Variable configuration.
- Time controls Time of Day's Day/Night Cycle automatically and smoothly (with updated script shown below).
- This works perfectly for Single or Multiplayer since everyone accesses the same Game Data.
- Super Easy to set up Calendar Based Gaming Events.
- Weather is also Persistently Stored....press play or the server loads up and weather returns exactly where it was.
- gDatabase.cs is set up so that it can be easily modified for SQL, SQLite, iBoxDB...or any other without modifying anything in other scripts. Please see this script for details.
- To figure out where your Databases are being stored....open gTime.cs and uncomment at line 39 //print(Application.persistentDataPath); Press Play then check your console.

Note: To change time in Edit or Play Mode click on \_GameData => gTime edit the N type variables. They automatically update as you enter/change data.

## Current Bugs and Issues:

- Wind is not working (to be honest I do not understand it yet :) ). Script and code placeholders are in place though.

## Giving Back to the Community:

- If you see an improvement to this weather system please give back to our community to make this very good weather system even better.
- If you have ideas post them here on this forum. We can figure out how to make it work.

**Preconfiguration Step:**

- Use the Unity Sample Project to get going fast.
- Time of Day v3.0.2 set up and functional (see those docs)
- Helps to have your camera move and/or look around. Plenty of options in default Unity.
- Camera - Global Fog Script Component (if converting a project from 4.x then remove ALL effects then reimport Assets-Import Package-Effects..5.0 is different)
- Persistency Database (for Time and Weather game data)...I recommend Easy Save 2 (the included database script is already setup for it...Plug and Play out of the box!! :) ) SQL, SQLite, and iBoxDB all work as well. You will need to reconfigure and test for those methods.
- \*If you do not have Easy Save 2 you will need to manually comment out any ES2 references in gOlanderWeather.cs => gDatabase.cs there are only a few.
- Unity 5 Free or Pro is ready to play. I have Unity 5 Pro and running it with DirectX11 and X9. This should also work in Unity 4.6 as well although not tested.

**Installation and Configuration: Step 1**

- Import OlandersRealisticWeather Unity Package
- Click on Olander's Realistic Weather Folder
- Drag \_GameData into Hierarchy...set to 0,0,0
- Drag \_WeatherMaster into Hierarchy...set to 0,0,0
- Drag \_WeatherObject into Hierarchy...set to 0,0,0 THEN into your Player/Moveable Object...animated character...not the camera. Set to 0,0,0

**Installation and Configuration: Step 2**

- CTRL+D TOD\_Time.cs rename to TOD\_TimeOrig.cs
- Open TOD\_Time.cs
- Line 10ish (just under => public class TOD\_Time : MonoBehaviour)

Copy and Paste:

```
//Added by Olander
    gTime gTime;

    /// Persistent Time
    /// This allow Time of Day to be controlled by Outside Time Systems
    /// Time of Day Curves may still be used in either option
    /// nUsePersistentTime :: 0 = Time of Day Time, 1 = Persistent Time
    [Tooltip("Persistent Time. nUsePersistentTime :: 0 = Time of Day Time, 1 = Persistent Time")]
    public int nUsePersistentTime = 1;
```

Line 233ish below the entire function => protected void Awake()

Copy and Paste this OVER what is there:

```
//Added by Olander
void Start ()
{
    //Get Time Data in _GameData
    gTime = GameObject.Find ("_GameData").GetComponent<gTime>();
}

protected void FixedUpdate()
{
    //Progress Time or Not (via Server/Database)
    //If not using this feature then comment out the 2nd line and Uncomment the 1st
    //bool ProgressTime = true;
    bool ProgressTime = gTime.bTimeProgress;

    //If Persistent Time is Enabled then Process Calculation
    if(nUsePersistentTime == 1)
    {
        //From gFunctions.cs Persistent Time Controller
        DayLengthInMinutes = gTime.fDayLengthMinutes;

        sky.Cycle.Hour = gTime.fHourCurrent;
        sky.Cycle.Day = gTime.nDayCurrent;
        sky.Cycle.Month = gTime.nMonthCurrent;
        sky.Cycle.Year = gTime.nYearCurrent;
    }
    else
    {
        //Time of Day Original
        if (ProgressTime && DayLengthInMinutes > 0)
        {
            const float oneDayInMinutes = 60 * 24;

            float timeFactor = oneDayInMinutes / DayLengthInMinutes;

            AddSeconds(Time.deltaTime * timeFactor);
        }
    }
}
```

- Save and Let Unity Compile

**Installation and Configuration: Step 3**

- Press Play....everything should be working. Now to have some fun. Return to Edit Mode.
- Scene Mode.....click on WeatherMaster. You will see all the configurations that may be toyed with. Leave them be but move your mouse over each variable to see the tooltip. They are quite detailed and tell you what each does.
- Olander! What is with all the F, N, S, G prior to the variables and scripts?? This makes it a simple matter to tell what should be in the variable...F-Float, N-Integer, B-Bool, S-String, G-GenesisEngine Script, C32-Color32 Unity Format....ect this is something 'Old School' C Language uses and I used this in Neverwinter Nights 1/2....works well for knowing what variable is formatted in which way. This method should be used by more programmers but I will say many new types have gotten used to the hover tools in software to see the variable....however....my argument still is....you still cannot tell what it is in Unity Inspector. \*wink\*
- \*\*end rant\*
- From Top to Bottom is the Flow you will need to configure each variable. For now....Press Play.....we will play with Humidity directly.

**Getting a feel of Humidity (Climate) Control: Step 4**

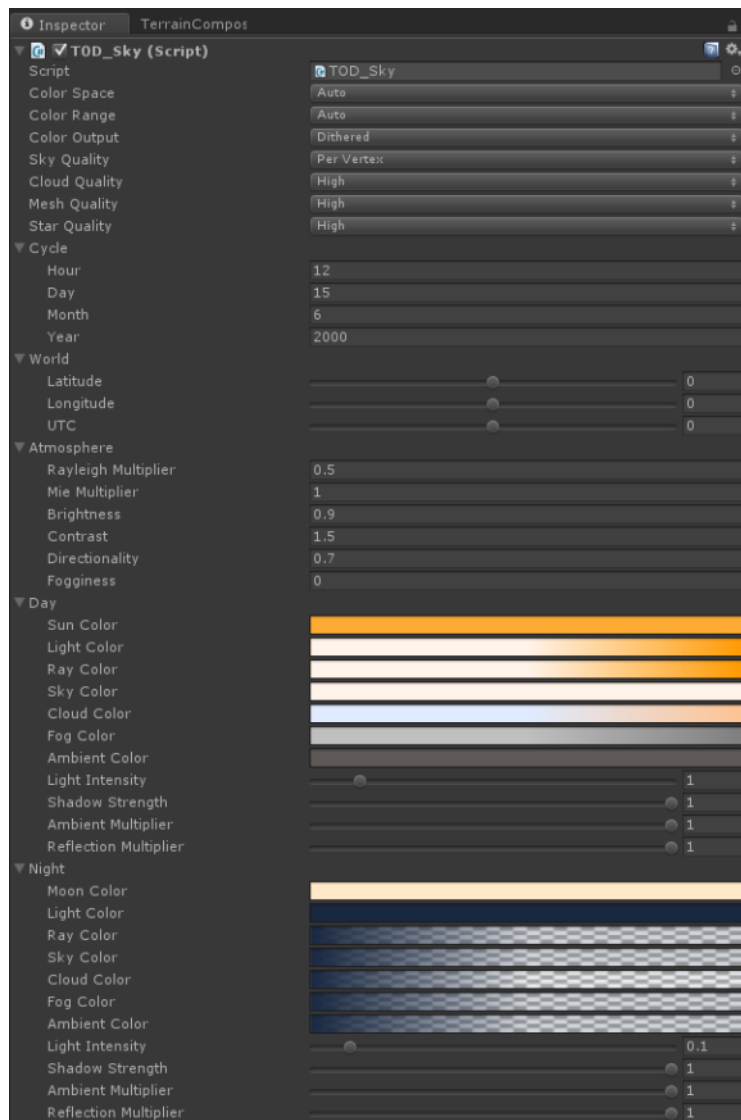
- You may notice that I have set a 10.0f second cycle rate. This feels very good but adjust it how you like once you get a feel for how weather works.
- Slowly increase Humidity and watch what happens to the Cloud Density and Cloud Brightness. As Humidity Increases clouds Thicken and become Darker.
- You will also notice that Global Fog is becoming ever dense as humidity increases. Add some Bloom to the camera for amazing results.
- Keep fTemperatureC at 25 for now. Fahrenheit below is automatically calculated.













- Set Humidity to 40 or higher. Then scroll up to => *nInColdFront* & *nInWarmFront* these are the current weather fronts. Cold Fronts condense air and precipitate while Warm Front absorb moisture and increase humidity. *nFrontDurationLeft* (FD) is a random length of 10 second cycles where a current front will persist. Once the front duration hits 0 and new front is determined and thus goes the cycle.
- For now...manually configure a Cold Front....set *nInWarmFront* to 0 and *nInColdFront* to 1. If Humidity is 50.0f or higher precipitation will start and continue until the Cold Front has passed (or less than 20% (configurable) humidity). To manually set a front duration simply set *nFrontDurationManualAdjust* to 1 then set the FD to what you would like. When FD has subtracted 1 then set *nFrontDurationManualAdjust* back to 0. Now the front duration is saved.
- Once you have the rain coming slowly increase humidity and watch the rain intensity increase with the clouds. The Weather Level (*nLevel*) controls the zonal weather type. As you increase from 1 to 6 weather also intensifies. You will see Thunder, Lightning Flashes, and Lightning Strikes begin to happen as well as the rain intensity is very wet. Play with this awhile....this weather system is enjoyable to play with but does take some time.
- Play with Sleet (between 0-1.5 Celsius) and then to Snow (less than 0 C). You should be feeling very comfortable playing with the variables at this point.

#### Playing with Effects Testing: Step 5

- A nice feature of Olander's Realistic Weather is you do not have to sit around to wait for random things to happen. Anything can be accessed by script, very easily. While in Play Mode go ahead and click on *\_Lightning*. See those B Test buttons? Click one. Ta Da. For Lightning you may need to click a couple of times since it is random locations and can be behind the camera. Now head over to *\_Thunder*..B Test Audio & N Test Type. Click the button to test. You will see the 11 second timer climb for the Thunder Sound. Now set the Type to say 5...and test. There are 9 types of thunder to listen to. If you set the type and click the test button it will wait until the 11 seconds has completed then automatically play the next test. Now head on over to *\_Rain*. Rain 1 and Rain 2 are the same audio (but could be different) but are set at two different volumes. Set F Rain Vol to 0.25 then click the Test Button. You can make sounds adjustments however you like then write down what you think is best when you return to edit mode. You can manually test the rain and snow particle effects by dragging a prefab into a scene and playing with that instead. The effects in the weather object are slaves to humidity and scripts....hence...they automatically controlled.

#### Time of Day Helpful Setup Pics:



▼ Sun	
Mesh Size	1
Mesh Brightness	2
Mesh Contrast	1
▼ Moon	
Mesh Size	1
Mesh Brightness	2
Mesh Contrast	1
Halo Size	0.1
Halo Brightness	1
Position	Realistic
▼ Stars	
Size	1
Brightness	1
Position	Rotating
▼ Clouds	
Size	1.5
Opacity	1
Coverage	1
Sharpness	0.5
Attenuation	0.5
Saturation	0.5
Scattering	1
Brightness	1.5
▼ Light	
Update Interval	0
Minimum Height	0
▼ Fog	
Mode	Color
Height Bias	0
▼ Ambient	
Mode	Color
Saturation	1
Update Interval	1
▼ Reflection	
Mode	None
Clear Flags	Skybox
Culling Mask	Nothing
Time Slicing	All Faces At Once
Update Interval	1
▶  TOD_Animation (Script)	 
▶  TOD_Time (Script)	 
▶  TOD_Components (Script)	 
▶  TOD_Resources (Script)	 
<div>Add Component</div>	