Справка по описанию неба.

Содержание

1 Общий вид описания неба:

```
Sky( имя_неба )
{
     // Фон вьюпорта
     Background( имя _фона )
     {
          …
     }

     // Изменение фона вьюпорта в зависимости от чего-либо
     BackgroundController( REF:имя_фона VAR:переменная )
     {
          …
     }

     // Туман
     Fog( имя_тумана )
     {
          …
     }

     // Изменение параметров тумана в зависимости от чего-либо
     FogController( REF:имя_тумана VAR:переменная )
     {
          …
     }

     // Дальняя плоскость отсечения
     FarClippingPlane( имя_плоскости )
     {
          …
     }

     // Изменение параметров дальней плоскости отсечения
     FarClippingPlaneController( REF:имя_плоскости VAR:переменная )
     {
          …
     }

     // Атмосфера, т.е. воздух без облаков и прочих эффектов
     Atmosphere( имя_атмосферы )
     {
          …
     }
```

```
// Изменение параметров атмосферы в зависимости от чего-либо
AtmosphereController( REF:имя_атмосферы VAR:переменная )
{
     …
}

// Небесное тело (солнце, луна)
Body( имя_тела )
{
     …
}

// Изменение параметров небесного тела в зависимости от чего-либо
BodyController( REF:имя_тела VAR:переменная )
{
     …
}


// Слой облаков (или любой меш, привязанный к камере)
CloudLayer( имя_слоя_облаков )
{
     …
}

// Изменение параметров слоя облаков в зависимости от чего-либо
CloudLayerController( REF:имя_слоя_облаков VAR:переменная )
{
     …
}


// Список дополнительных pml-файлов
MatLibIni( имя_секции )
{
     …
}

// Изменение параметров материала в зависимости от чего-либо
MaterialController( REF:имя_слоя_облаков VAR:переменная )
{
     …
}

// Источник звука
Sound( имя_источника_звука )
{
     …
}

// Изменение параметров источника звука в зависимости от чего-либо
```

```
    SoundController( REF:имя_источника_звука VAR:переменная )
    {
        …
    }

    // Оверлей (дополнительное небо, накладываемое на это небо)
    Overlay( имя_оверлея )
    {
        …
    }

    // Изменение веса оверлея в зависимости от чего-либо
    OverlayController( REF:имя_оверлея VAR:переменная )
    {
        …
    }

    // Функция времени (имеет вспомогательное значение)
    Function( имя_функции )
    {
        …
    }
}
```

Каждое небо имеет имя, которое записывается в заголовке. Имя неба должно совпадать с именем файла *.sky, в котором это небо определено.

Описание неба состоит из секций; каждая секция описывает свой аспект неба. Каждая секция имеет имя. Регистр букв в файле *.sky (sky-скрипте) не учитывается.

Если в zen-мире нет воба типа ZoneSkyDefault, то используется небо по умолчанию с тем же именем, что и имя мира, например, OldWorld.sky. Если такого файла тоже не найдено, то используется DefaultOutdoor.sky или DefaultIndoor.sky, в зависимости от типа локации (типа мира).

---

2 Типы данных

- **Строка.**

текстовые строки записываются в кавычках

- **Время**.

Время может задаваться по-разному:
"0.3" - 0.3 секунды;
"5" - 5 секунд;
"20:30" - 20 часов, 30 минут;
"20:30:05" - 20 часов, 30 минут, 5 секунд;
"0:30:05.001" - 30 минут, 5 секунд, 1 миллисекунда;
"5 days, 5:15" - пять дней, 5 часов и 15 минут;

"Day 5, 5:15" - то же.

При задании времени можно дополнительно задать период, т.е. время, по прошествии которого событие будет повторено (солнце встает каждое утро и т.п.). Для этого используется ключ PERIOD. Если период не задан (или задан ноль), то он считается равным +Infinity, т.е. событие никогда не будет повторяться.

Примеры:
”Day 3, 6:30” - полседьмого утра, в третий день;
”Day 3, 6:30” PERIOD:”24:00” - каждый день полседьмого утра, начиная с третьего дня;
”Day 3, 6:30” PERIOD:”1 day” - то же,
”6:30” PERIOD:”24:00” - каждый день полседьмого утра;
”0:00” PERIOD:”2:00” - каждые два часа, начиная с полуночи.

- **Углы и расстояния.**

Все углы задаются в градусах, все расстояния - в сантиметрах

- **Цвет.**

записываются три или четыре компоненты, в кавычках:
"R G B A"
каждая компонента указывается как целое число от 0 до 255.
A(Alpha) - степень непрозрачности (0 - полностью прозрачно, 255 - полностью непрозрачно).
Если указано всего три компоненты цвета, то последняя (A) подразумевается равной 255.

Пример:
"150 160 78 200"

---

3 Модификаторы

Модификатор изменяет текущее значение какой-либо переменной.

- **Модификатор расстояния**

модификатор расстояния может преобразовать какое-либо расстояние

Варианты записи модификатора:
ADD:5800 - добавить 5800 см

MUL:1.3 - умножить на 1.3:

SET:9000 — установить расстояние 9000 см.
Слово SET можно опустить, т.е. записать просто
9000

Можно комбинировать несколько вариантов модификаторов в одной записи:
MUL:1.1 ADD:4000

Порядок имеет значение:
MUL:1.1 ADD:4000

-не то же самое, что
ADD:4000 MUL:1.1

- **Модификатор угла.**

Записывается аналогично модификатору расстояния

- **Модификатор цвета**

модификатор цвета может преобразовать цвет

Варианты записи модификатора:
ADD:"58 30 90 11" - добавить указанный цвет к исходному цвету

MUL:"111 222 90 30" - умножить исходный цвет на указанный цвет, формула:
результат.R = исходный_цвет.R * (множитель.R / 255.0)
результат.G = исходный_цвет.G * (множитель.G / 255.0)
результат.B = исходный_цвет.B * (множитель.B / 255.0)
результат.A = исходный_цвет.A * (множитель.A / 255.0)

Т.е. MUL работает как фильтр, оставляя от исходного цвета лишь указанную часть, например:
MUL:"0 255 0 255" (или MUL:"0 255 0") - оставит лишь зеленый компонент.

SET:"90 89 67 23" - просто заменить исходный цвет указанным.
Слово SET можно опустить, т.е. записать просто
"90 89 67 23"

Можно комбинировать несколько вариантов модификаторов в одной записи:
MUL:"10 20 30" ADD:"40 50 60"

Порядок имеет значение:
MUL:"10 20 30" ADD:"40 50 60"
-не то же самое, что
ADD:"40 50 60" MUL:"10 20 30"

- **Модификатор вектора (углы эйлера, коэффициент масштабирования)**

модификатор вектора может преобразовать вектор

Варианты записи модификатора:
ADD:"100 2000 30000" - добавить указанный вектор

MUL:"1.1 2.2 3.3" - умножить каждый компонент вектора на соответствующее число; это не скалярное и не векторное произведение, а просто покомпонентное перемножение:
результат.X = исходный.X * множитель.X
результат.Y = исходный.Y * множитель.Y
результат.Z = исходный.Z * множитель.Z

SET:"900 100 -1111" - просто заменить текущее значение на указанное.
Слово SET можно опустить, т.е. записать просто
"900 100 -1111"

Можно комбинировать несколько вариантов модификаторов в одной записи:
MUL:"0.1 0.7 2.2" ADD:"100 2000 30000"

Порядок имеет значение:
MUL:"0.1 0.7 2.2" ADD:"100 2000 30000"
-не то же самое, что
ADD:"100 2000 30000" MUL:"0.1 0.7 2.2"

- Индекс слоя

Индекс слоя - это целое число от 1 до 19.
Элементы неба рендерятся в порядке возрастания индекса слоя.
Рекомендуется использовать такие индексы слоев:
1 - звезды
3 - луна
5 - ночные облака №1
6 - ночные облака №2
10 - атмосфера (слои с индексами ниже индекса слоя атмосферы днем будут не видны)
11 - солнце
12 - дневные облака №1
13 - дневные облака №2

---

4 Контроллеры

Секции, название которыз заканчивается словом “Controller”, задают изменение параметров аспекта неба, описанного ранее. Изменение параметров может быть:

1) Безусловным, т.е. так:

```
FogController( REF:имя_тумана )
{
     color( модификатор_цвета )
     nearDistance( модификатор_расстояния )
     farDistance( модификатор_расстояния )
}
```

2) Условным, т.е. зависимым от какой-либо переменной. В этом случае используется ключ VAR:, чтобы указать, от чего должно зависеть изменение параметров. Условный контроллер может быть:

- **Зависящим от времени**

```
FogController( REF:имя_тумана VAR:WORLD_TIME )
{
     at( время )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     at( время )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
```

```
            farDistance( модификатор_расстояния )
      }
      …
}
```

Указываются модификаторы параметров тумана для некоторых моментов времени; для остальных моментов времени эти параметры интерполируются (линейная интерполяция).

- **Зависящим от времени по более сложным формулам — через специальную функцию (определенную в секции FUNCTION):**

```
FogController( REF:имя_тумана VAR:FUNCTION:имя_функции )
{
      at( значение_функции )
      {
            color( модификатор_цвета )
            nearDistance( модификатор_расстояния )
            farDistance( модификатор_расстояния )
      }
      at( значение_функции )
      {
            color( модификатор_цвета )
            nearDistance( модификатор_расстояния )
            farDistance( модификатор_расстояния )
      }
      …
}
```

- **Зависящим от высоты над горизонтом указанного небесного тела:**

```
FogController( REF:имя_тумана VAR:ALTITUDE_OF_BODY:имя_тела )
{
      at( высота_тела_над_горизонтом )
      {
            color( модификатор_цвета )
            nearDistance( модификатор_расстояния )
            farDistance( модификатор_расстояния )
      }
      at( высота_тела_над_горизонтом )
      {
            color( модификатор_цвета )
            nearDistance( модификатор_расстояния )
            farDistance( модификатор_расстояния )
      }
      …
}
```

Примечание: Высота тела над горизонтом изменяется в интервале от -90 до 90 градусов.

- **Зависящим от глубины под водой:**

```
FogController( REF:имя_тумана VAR:UNDERWATER_DEPTH )
```

```
{
     at( глубина )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     at( глубина )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     …
}
```

Примечание: Если камера находится над водой, глубина считается отрицательной.

- **Зависящим от глубины внутри помещения:**

```
FogController( REF:имя_тумана VAR:INDOOR_DEPTH )
{
     at( глубина )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     at( глубина )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     …
}
```

Примечание: Если камера не находится внутри помещения, глубина считается отрицательной.

3) Комбинированным — т.е. часть параметров меняется безусловно, а часть — в зависимости от какой-либо переменной, например,

```
FogController( REF:имя_тумана VAR:WORLD_TIME )
{
     nearDistance( модификатор_расстояния )
     farDistance( модификатор_расстояния )
     at( время )
     {
          color( модификатор_цвета )
          ~~nearDistance( модификатор_расстояния )~~
     }
     at( время )
```

```
        {
            color( модификатор_цвета )
            ~~farDistance( модификатор_расстояния )~~
        }
        ~~color( модификатор_цвета )~~
        ...
    }
```

Здесь изменение nearDistance и farDistance не зависит от времени, а изменение цвета тумана — зависит. Перечеркнутые строки показывают, что если изменение nearDistance и farDistance описано как не зависящее от времени, то эти модификаторы появляться в блоке at не должны. Безусловные изменения параметров должны быть записаны раньше условных.

---

5 Время: начало отсчета и скорость хода

При задании зависимости от времени используется одна из форм:

- WORLD_TIME – мировое время, или время виртуального мира. Мировое время течет быстрее реального в 14.4 раза, но этот коэффициент может меняться — мировое время может ускоряться, замедляться и даже останавливаться (когда игру ставят на паузу). Начало отсчета мирового времени определяется в скриптах. Например, в начале игры обычно игровое время равно «8:00».
- WORLD_TIME_REL:имя_неба – тоже мировое время, но начало отсчета совпадает с тем моментом, когда вес указанного неба последний раз стал больше 0.
- SYSTEM_TIME – системное, или реальное, время — время реального мира, определенное по часам компьютера. Скорость хода реального времени неизменна. Таким образом, если игра запущена 14 мая в 13:40, то системное время в начале игры будет таким: «Day 14, 13:40”.
- SYSTEM_TIME_REL:имя_неба — тоже системное время, но начало отсчета совпадает с тем моментом, когда вес указанного неба последний раз стал больше 0.

Например,

```
FogController( REF:имя_тумана VAR:SYSTEM_TIME_REL:имя_неба )
{
    at( время )
    {
        color( модификатор_цвета )
        nearDistance( модификатор_расстояния )
        farDistance( модификатор_расстояния )
    }
    at( время )
    {
        color( модификатор_цвета )
        nearDistance( модификатор_расстояния )
        farDistance( модификатор_расстояния )
    }
    …
```

```
}
```

---

6 Background

Секция Background служит для задания цвета фона вьюпорта.
(Фон вьюпорта зачастую виден только в случае, когда он не заслонен звездным небом / атмосферой / облаками).

Общий вид записи:

```
Background( имя_фона )
{
     color( цвет )
}
```

В заголовке задается имя фона, которое потом может быть использовано для ссылок на это описание фона.

Если не определено ни одной секции Background, то будет использоваться черный цвет фона.
Если определено более одной секции Background, то будет использоваться последняя.

---

7 BackgroundController

Секция BackgroundController служит для изменения цвета фона вьюпорта.

```
BackgroundController( REF:имя_фона VAR:переменная )
{
     at( значение_функции )
     {
          color( модификатор_цвета )
     }
     at( значение_функции )
     {
          color( модификатор_цвета )
     }
     …
}
```

В заголовке задается имя фона, т.е. имя секции “Background”, параметры которой изменяются.

Если задано более одной секции BackgroundController с тем же REF:имя_фона, то каждое последующее описание накладывается на предыдущее (модификаторы цвета применяются в порядке описания секций).

8 Fog

Секция Fog описывает Z-туман, который перекрашивает объекты, удаленные от камеры, в цвет тумана.

Общий вид записи:

```
Fog( имя_тумана )
{
     color( цвет )
     nearDistance( расстояние )
     farDistance( расстояние )
}
```

В заголовке задается имя тумана, которое потом может быть использовано для ссылок на это описание тумана.

Параметры:

- color – задает цвет тумана
- nearDistance – ближнее расстояние тумана, т.е. расстояние, с которого начинается туман. Ближе этого расстояния тумана нет.
- farDistance – дальнее расстояние тумана, т.е. расстояние, где туман окончательно перекрашивает все объекты сцены в свой цвет.

Если не определено ни одной секции Fog, то туман использоваться не будет.
Если определено более одной секции Fog, то будет использоваться последняя.

9 FogController

Секция FogController служит для изменения параметров тумана.

```
FogController( REF:имя_тумана VAR:переменная )
{
     at( значение_функции )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     at( значение_функции )
     {
          color( модификатор_цвета )
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     ...
```

```
}
```

В заголовке задается имя тумана, т.е. имя секции “Fog”, параметры которой изменяются.

Допустимо указывать изменение не всех параметров, например,

```
FogController( REF:имя_тумана VAR:переменная)
{
     at( значение_функции )
     {
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     at( значение_функции )
     {
          nearDistance( модификатор_расстояния )
          farDistance( модификатор_расстояния )
     }
     ...
}
```

Если задано более одной секции FogController с тем же REF:имя_тумана, то каждое последующее описание накладывается на предыдущее (модификаторы цвета и расстояния применяются в порядке описания секций).

---

10 FarClippingPlane

Секция FarClippingPlane задает расстояние от камеры до дальней плоскости отсечения, таким образом вместе с туманом регулируя дальность видимости.

Общий вид записи:

```
FarClippingPlane( имя_плоскости )
{
     distance( расстояние )
}
```

Параметры:

- distance – расстояние до дальней плоскости отсечения, в сантиметрах.

Если не определено ни одной секции FarClippingPlane, то дальняя плоскость отсечения использоваться не будет (не будет ограничивать дальность видимости).
Если определено более одной секции FarClippingPlane, то будет использоваться последняя.

11 FarClippingPlaneController

Секция FarClippingPlaneController служит для изменения параметров дальней плоскости отсечения.

```
FarClippingPlaneController( REF:имя_плоскости VAR:переменная )
{
     at( значение_функции )
     {
          distance( модификатор_расстояния )
     }
     at( значение_функции )
     {
          distance( модификатор_расстояния )
     }
     ...
}
```

В заголовке задается имя плоскости, т.е. имя секции “FarClippingPlane”, параметры которой изменяются.

12 Atmosphere

Секция Atmosphere описывает атмосферу, т.е. то, как должен рисоваться воздух без облаков. Атмосфера рисуется как сфера с отсеченной нижней частью, накрывающая сверху наблюдателя.

Параметры атмосферы:

- radius – радиус сферы, в сантиметрах.
- minAltitude (от -90 до 0) задает минимальную высоту (в градусах), до которой рисуется атмосфера. Высота отсчитывается как принято в небесной механике, то есть 0 градусов — точка на горизонте, 90 градусов — в зените, -90 градусов — в надире. Таким образом, если задать minAltitude равным -90 градусов, то будет нарисована вся сфера, а если задать minAltitude равным 0 градусов, то будет нарисована ровно верхняя полусфера. Рекомендуется указывать значение, меньшее нуля, чтобы при прыжках и полетах не было видно нижнего края атмосферы. Предпочтительно, чтобы параметр minAltitude удовлетворял неравенству:

$$minAltitude \leqslant -\arcsin\left(\frac{maxFlightHeight}{radius}\right)$$

Общий вид записи:

```
Atmosphere( имя_атмосферы )
```

```
{
     geometry( RADIUS:радиус_полусферы minAltitude:мин_высота )
     layer( индекс_слоя )
     ambientLightFilter( цвет )
     diffuseLightFilter( цвет )
     specularLightFilter( цвет )
     ambientLight( цвет )
     color( цвет )
}
```

В заголовке указывается имя атмосферы, которое может быть в дальнейшем использовано для ссылок на эту атмосферу.

Параметры:

- geometry – задает геометрические размеры атмосферы: радиус сферы и минимальную высоту (см. описание выше).
- layer - задает индекс слоя атмосферы;
- color - задает цвет атмосферы (может быть изменен впоследствии - см. секции
- AtmosphereController).
- ambientLight — цвет рассеянного освещения от атмосферы.
- ambientLightFilter, diffuseLightFilter, specularLightFilter – задают фильтрацию света от источников, индекс слоя которых меньше индекса слоя атмосферы.

Например,

```
ambientLightFilter( "255 0 0")
diffuseLightFilter( "255 0 0")
specularLightFilter("255 0 0")
```

- атмосфера, пропускающая только красный канал света

Фактически цвет источников света, находящихся за атмосферой, умножается на цвет, заданный в качестве фильтра.

Значения по умолчанию:

```
ambientLight("0 0 0")
ambientLightFilter( "255 255 255")
diffuseLightFilter( "255 255 255")
specularLightFilter("255 255 255")
color("0 0 255 255")
```

дадут синию непрозрачную атмосферу, без собственного свечения, без фильтрации.

---

13 AtmosphereController

Секция AtmosphereController используется для изменения параметров атмосферы.

Общий вид записи:

```
AtmosphereController( REF:имя_атмосферы VAR:переменная )
{
```

```
     at( значение_функции )
     {
          ambientLightFilter( модификатор_цвета )
          diffuseLightFilter( модификатор_цвета )
          specularLightFilter( модификатор_цвета )
          ambientLight( модификатор_цвета )
          color( модификатор_цвета )
     }
     at( значение_функции )
     {
          ambientLightFilter( модификатор_цвета )
          diffuseLightFilter( модификатор_цвета )
          specularLightFilter( модификатор_цвета )
          ambientLight( модификатор_цвета )
          color( модификатор_цвета )
     }
     ...
}
```

Можно задавать не все перечисленные параметры, а только часть.

Если задано более одной секции AtmosphereController с тем же REF:имя_атмосферы, то каждое последующее описание накладывается на предыдущее (модификаторы цвета применяются в порядке описания секций).

---

14 AtmosphereController (VAR:ALTITUDE_OF_BODY)

При задании зависимости параметров атмосферы от перемещения небесного тела можно также задать изменение цвета в отдельных точках атмосферы, а не сразу для всей атмосферы.

Общий вид записи:

```
AtmosphereController( REF:имя_атмосферы VAR:ALTITUDE_OF_BODY:имя_тела)
{
     at( высота_тела_над_горизонтом )
     {
          ambientLightFilter( модификатор_цвета )
          diffuseLightFilter( модификатор_цвета )
          specularLightFilter( модификатор_цвета )
          ambientLight( модификатор_цвета )
          at угол_между_телом_и_точкой color( модификатор_цвета )
          at угол_между_телом_и_точкой color( модификатор_цвета )
          …
     }
     at(  высота_тела_над_горизонтом )
     {
          ambientLightFilter( модификатор_цвета )
          diffuseLightFilter( модификатор_цвета )
```

```
            specularLightFilter( модификатор_цвета )
            ambientLight( модификатор_цвета )
            at угол_между_телом_и_точкой color( модификатор_цвета )
            at угол_между_телом_и_точкой color( модификатор_цвета )
            …
      }
      …
}
```

Строки вида
`at угол_между_телом_и_точкой color( модификатор_цвета )`
задают модификацию цвета самой атмосферы в точке, направление на которую образует с направлением на небесное тело указанный угол. Для тех значений углов, для которых не указаны модификаторы цвета, эти модификаторы вычисляются с использованием (линейной) интерполяции.

---

15 Body

Секция Body служит для описания небесного тела (солнца, луны, и т.п.), или любого меша, чьи координаты определяются высотой и азимутом.

Общий вид записи:

```
Body( имя_тела )
{
      visual( FILE:имя_3ds_файла RESOURCE_GROUP:группа_ресурсов MATERIAL:имя_материала )
      layer( индекс_слоя )
      altitude( высота )
      azimuth( азимут )
      distance( расстояние )
      scale( коэффициент_масштабирования )
      eulerAngles( углы_Эйлера )
      position( смещение )
      ambientLight( цвет )
      diffuseLight( цвет )
      specularLight( цвет )
}
```

В заголовке задается имя небесного тела. Это имя может использоваться для ссылок на это тело.

Параметры:

- visual – задает имя файла, из которого нужно загрузить меш. Группу ресурсов можно не указывать, т.к. значение по умолчанию - “MESHES” - вполне подходит в большинстве случаев. Также можно дополнительно задать материал, который должен быть использован для меша, вместо материала из файла.
- layer – задает индекс слоя, который влияет на порядок рендеринга

- altitude – задает высоту небесного тела над горизонтом, [-90..90].
- azimuth – задает азимут небесного тела (угол между направлением на север и горизонтальным направлением в сторону небесного тела). [-180..180].
- distance – расстояние (от камеры), на котором помещается меш небесного тела.
- scale – коэффициент маштабирования (вектор из трех чисел); этот параметр вместе с параметром distance влияет на видимый размер небесного тела.
- eulerAngles – углы Эйлера (вектор из трех чисел), задают поворот небесного тела относительно своей оси.
- position – дополнительное смещение небесного тела относительно позиции, рассчитанной по высоте и азимуту
- ambientLight – цвет рассеянного освещения от данного небесного тела
- diffuseLight – цвет диффузного освещения от данного небесного тела
- specularLight – цвет бликового освещения от данного небесного тела.

Фактически цвет источников света, находящихся за атмосферой, умножается на цвет, заданный в качестве фильтра.

Значения по умолчанию:
```
altitude( 90 )
azimuth( 0 )
distance( 10000 )
scale( "1 1 1" )
eulerAngles( "0 0 0" )
position( "0 0 0" )
ambientLight( "0 0 0" )
diffuseLight( "0 0 0" )
specularLight( "0 0 0" )
```
- означают, что небесное тело находится в зените, на расстоянии 10000см (100м) от наблюдателя, при этом света не дает.

---

16 BodyController

Секция BodyController используется для изменения параметров небесного тела.

Общий вид записи:

```
BodyController( REF:имя_тела VAR:переменная )
{
     at( значение_функции )
     {
          altitude( модификатор_высоты )
          azimuth( модификатор_азимута )
          distance( модификатор_расстояния )
          scale( модификатор_коэффициента_масштабирования )
          eulerAngles( модификатор_углов_Эйлера )
          position( модификатор_смещения )
          ambientLight( модификатор_цвета )
```

```
            diffuseLight( модификатор_цвета )
            specularLight( модификатор_цвета )
      }
      at( значение_функции )
      {
            altitude( модификатор_высоты )
            azimuth( модификатор_азимута )
            distance( модификатор_расстояния )
            scale( модификатор_коэффициента_масштабирования )
            eulerAngles( модификатор_углов_Эйлера )
            position( модификатор_смещения )
            ambientLight( модификатор_цвета )
            diffuseLight( модификатор_цвета )
            specularLight( модификатор_цвета )
      }
      ...
}
```

Можно задавать не все перечисленные параметры, а только часть.

Если задано более одной секции BodyController с тем же REF:имя_тела, то каждое последующее описание накладывается на предыдущее (модификаторы цвета применяются в порядке описания секций).

---

17 CloudLayer

Секция CloudLayer описывает слой облаков, или любой меш, чья позиция привязывается к позиции камеры.

Общий вид записи:

```
CloudLayer( имя_слоя_облаков )
{
      visual( FILE:имя_3ds_файла RESOURCE_GROUP:группа_ресурсов MATERIAL:имя_материала )
      layer( индекс_слоя )
      position( позиция )
      eulerAngles( углы_Эйлера )
      scale( коэффициент_масштабирования )
      ambientLightFilter( цвет )
      diffuseLightFilter( цвет )
      specularLightFilter( цвет )
      ambientLight( цвет )
      diffuseLight( цвет )
      specularLight( цвет )
}
```

В заголовке указывается имя атмосферы, которое может быть в дальнейшем использовано для ссылок на эту атмосферу.

Параметры:

- visual – задает имя файла, из которого нужно загрузить меш. Группу ресурсов можно не указывать, т.к. значение по умолчанию - “MESHES” - вполне подходит в большинстве случаев. Также можно дополнительно задать материал, который должен быть использован для меша, вместо материала из файла.
- layer - задает индекс слоя, который влияет на порядок рендеринга;
- position - задает позицию меша относительно камеры;
- scale – задает коэффициент масштабирования (вектор из трех чисел) меша;
- eulerAngles – задает углы Эйлера, указывающие ориентацию меша;
- ambientLightFilter, diffuseLightFilter, specularLightFilter – задают фильтрацию света от источников, индекс слоя которых меньше индекса слоя облаков
- ambientLight — цвет рассеянного освещения от облаков (облака могут светиться);
- diffuseLight, specularLight – цвета диффузного и бликового освещения, производимого источником света, помещенным в позицию, заданную параметром position; эти цвета почти всегда нулевые, так как с облаками обычно не должен быть связан точечный источник света.

Например,
```
ambientLightFilter( “255 0 0”)
diffuseLightFilter( “255 0 0”)
specularLightFilter(“255 0 0”)
```
- облака, пропускающие только красный канал света

Значения по умолчанию:
```
position( “0 0 0” )
scale( “1 1 1”)
eulerAngles( “0 0 0” )
ambientLightFilter( “255 255 255”)
diffuseLightFilter( “255 255 255”)
specularLightFilter( “255 255 255” )
ambientLight( “0 0 0” )
diffuseLight( “0 0 0” )
specularLight( “0 0 0” )
```

---

18 CloudLayerController

Секция CloudLayerController позволяет изменять параметры слоя облаков.

```
CloudLayerController( имя_слоя_облаков VAR:переменная)
{
     at( значение_функции )
     {
          position( модификатор_позиции )
          eulerAngles( модификатор_углов_Эйлера )
          scale( модификатор_коэффициента_масштабирования )
          ambientLightFilter( модификатор_цвета )
          diffuseLightFilter( модификатор_цвета )
          specularLightFilter( модификатор_цвета )
```

```
            ambientLight( модификатор_цвета )
            diffuseLight( модификатор_цвета )
            specularLight( модификатор_цвета )
      }
      at( значение_функции )
      {
            position( модификатор_позиции )
            eulerAngles( модификатор_углов_Эйлера )
            scale( модификатор_коэффициента_масштабирования )
            ambientLightFilter( модификатор_цвета )
            diffuseLightFilter( модификатор_цвета )
            specularLightFilter( модификатор_цвета )
            ambientLight( модификатор_цвета )
            diffuseLight( модификатор_цвета )
            specularLight( модификатор_цвета )
      }
      …
}
```

---

19 MatLibIni

Секция MatLibIni позволяет задать список pml-файлов, которые содержат материалы, которые будут загружены при инициализации этого неба.

Общий вид записи:

```
MatLibIni( имя_секции_не_используется )
{
      pmlFile( имя_файла )
      pmlFile( имя_файла )
      pmlFile( имя_файла )
      ...
}
```

---

20 MaterialController

Секция MaterialController позволяет изменять материал. Изменение материала приводит к изменению внешнего вида всех полигонов мира, на которые этот материал натянут.

Общий вид записи:

```
MaterialController( REF:имя_материала VAR:переменная)
```

```
{
      at( значение_функции )
      {
            frame( номер_кадра )

            texAniMapDir( скорость_скроллинга_по_гориз_и_по_вертикали )
            colorFactor( множитель_для_диффузного_цвета )

            specularIntensityFactor( множитель_для_яркости_бликового_пятна )

            specularColorFactor( множитель_для_цвета_бликового_пятна )

            reflectivityFactor( множитель_для_доли_отраженного_изображения )

      }
      at( значение_функции )
      {
            frame( номер_кадра )

            texAniMapDir( скорость_скроллинга_по_гориз_и_по_вертикали )
            colorFactor( множитель_для_диффузного_цвета )

            specularIntensityFactor( множитель_для_яркости_бликового_пятна )

            specularColorFactor( множитель_для_цвета_бликового_пятна )

            reflectivityFactor( множитель_для_доли_отраженного_изображения )

      }
      …
}
```

Можно задавать не все перечисленные параметры, а только часть.

Если задано более одной секции MaterialController с тем же REF:имя_материала, то каждое последующее описание накладывается на предыдущее (модификаторы применяются в порядке описания секций).

---

21 Sound

Секция Sound служит для определения звука, воспроизводимого небом (типа звуков дождя и т.п.)

Общий вид записи:

```
Sound( имя_звука )
{
      dataSource( SFX:имя_инстанции | FILE:имя_файла RESOURCE_GROUP:группа )
      looping( зацикливать_или_нет )
      position( позиция )
      maxDistance( расстояние )
      volume( громкость )
```

```
}
```

В заголовке задается имя звука, которое может использоваться затем для ссылок на этот звук.

Параметры:

- dataSource – источник данных, указывает, где брать звуковые данные. Нужно указывать либо SFX:имя_инстанции (в sfx.dat), либо имя звукового файла вместе с именем ресурсной группы. Имя ресурсной группы (“SOUND” или “MUSIC”) иопределяет, где искать этот звуковой файл.
- looping – указывает, нужно ли зацикливать звук (0 — не зацикливать, 1 — зацикливать).
- position – позиция звука относительно камеры
- maxDistance – максимальное удаление от источника звука, на котором звук езще слышен.
- volume – громкость звука, в процентах (0 — тишина, 100 — максимум).

Значения по умолчанию:

```
looping( 0 )
position( “0 0 0” )
maxDistance( “1” )
volume( “100” )
```

---

22 SoundController

Секция SoundController позволяет задать изменение источника звука в зависимости от чего-либо.

```
SoundController( REF:имя_звука VAR:переменная)
{
     at( значение_функции )
     {
          volume( модификатор_громкости )
          position( модификатор_позиции )
          maxDistance( модификатор_расстояния )
     }
     at( значение_функции )
     {
          volume( модификатор_громкости )
```

```
        position( модификатор_позиции )
        maxDistance( модификатор_расстояния )
    }
    ...
}
```

---

23 Overlay

Оверлей — это дополнительное небо, которое должно накладываться на это небо. При этом:

- элементы неба-оверлея сортируются по слоям вместе с элементами основного неба
- порядок применения модификаторов: 1) вначале применяются модификаторы основного неба, независимо от того, описаны они до оверлея или после; 2) потом модификаторы оверлеев, в порядке описания

Общий вид записи оверлея:

```
Overlay( имя_оверлея )
{
    sky( имя_неба )
    weight( вес_неба )
}
```

Параметры:

- sky – имя неба (т.е. имя файла с расширением .sky), содержащего эффекты, требуемые оверлею.
- weight – вес неба-оверлея, показывающий, в какой степени должны использоваться эффекты оверлея (0.0 — отсутствие эффекта, 1.0 — полный эффект). Например, если вес равен 0.3, то: звук из неба-оверлея звучит на 30% своей нормальной громкости, небесное тело из неба-оверлея светит на 30% яркости и вдобавок лишь на 30% непрозрачно, слой облаков из неба-оверлея на 30% непрозрачен, и т.д.

---

24 OverlayController

Секция OverlayController позволяет задать изменение веса оверлея.

Общий вид записи:

```
OverlayController( REF:имя_оверлея VAR:переменная )
{
```

```
    at( значение_функции )
    {
        weight( вес_неба )
    }
    at( значение_функции )
    {
        weight( вес_неба )
    }
    …
}
```

---

25 Function

Секция Function служит для описания сложной функции времени, которая может быть полезна для описания природных явлений.

Общий вид записи:

```
Function( имя_функции )
{
    type( тип_функции )
    argument( WORLD_TIME | WORLD_TIME_REL:имя_неба | SYSTEM_TIME | SYSTEM_TIME_REL:имя_неба )
    …
}
```

Параметры:

- type – задает тип функции, который определяет, как именно функция будет генерировать значения. Этот параметр является обязательным и не может быть пропущен.
- argument – тип времени, от которого зависит данная функция. По умолчанию используется WORLD_TIME.

Тип функции может быть одним из следующих:

- **CONSTANT** – постоянная функция. Все время равна одному и тому же значению.

  Дополнительные параметры:

  value(значение)

  Формула:

  f(t) = value

  Пример:

  ```
  Function( MyFunction )
  ```

```
{
    type( CONSTANT )
    value( 50 )
}
```

- **LINEAR** — линейная функция.

  Дополнительные параметры:

  initValue(начальное_значение)

  rate(прирост_значения_за_1_секунду)

  period(период)

  Формула (если период не задан или равен нулю):

  $f(t) = rate \cdot t + initValue$,

  где t – время.

  Формула (если период задан):

  $f(t) = rate \cdot t_0 + initValue$,

  где $t_0 = t - floor(t / period) \cdot period$ – время относительно этого периода.

  Пример:

```
Function( MyFunction )
{
    type( LINEAR )
    initValue( 100 )
    rate( -1 )
    argument( WORLD_TIME )
    period( 12 )
}
```

- **QUADRATIC** – квадратичная функция.

  Дополнительные параметры:

  initValue(начальное_значение)

  initRate(начальный_прирост)

  rate2(прирост_прироста_за_1_секунду)

  period(период)

  Формула (если период не задан или равен нулю):

$$f(t)=\frac{rate2\cdot t^2}{2}+initRate\cdot t+initValue \quad .$$

Формула (если период задан):

$$f(t)=\frac{rate2\cdot t_0^2}{2}+initRate\cdot t_0+initValue \quad ,$$

где $t_0=t-floor(t/period)\cdot period$ – время относительно этого периода.

Пример:

```
Function( MyFunction )
{
     type( QUADRATIC )
     initValue( 7 )
     initRate( 34 )
     rate2( 10 )
}
```

- **SIN** – гармоническая зависимость.

Дополнительные параметры:

minValue(минимальное_значение)

maxValue(максимальное_значение)

period(период)

initPhase(начальная_фаза)

Формула:

$$f(t)=\frac{maxValue-minValue}{2}\cdot\sin\left(\frac{2\cdot\pi\cdot t}{period}+initPhase\right)+\frac{minValue+maxValue}{2}$$

Пример:

```
Function( MyFunction )
{
     type( SIN )
     minValue( 10 )
     maxValue( 20 )
     period( 200 )
```

```
        initPhase( 0 )
    }
```

- **RANDOM_VALUE** — случайная функция, каждое следующее значение которой определяется генератором случайных чисел. То есть значение этой функции в следующий момент времени никак не связано со значением в предыдущий момент. Совершенная непредсказуемость.

  Дополнительные параметры:

  minValue(минимальное_значение)

  maxValue(максимальное_значение)

  step(шаг)

  Формулы:

  $$f(0) = random(minValue, maxValue)$$

  $$f(t + \Delta t) = f(t) \quad \text{если} \quad \Delta t < step$$

  $$f(t + step) = random(minValue, maxValue)$$

  Здесь $random(a, b)$ — функция, генерирующая случайное число от a до b.

  То есть параметр step задает интервал времени, через который генератор случайных чисел должен использоваться снова.

  Обозначения: t – текущий момент времени, $t + \Delta t$ – следующий момент времени, т.е. следующее значение функции вычисляется через предыдущее.

  Пример (определение случайной функции, генерирующей значения от 50 до 80):

```
    Function( MyFunction )
    {
        type( RANDOM_VALUE )
        minValue( 50 )
        maxValue( 80 )
        step( 1 )
    }
```

- **RANDOM_RATE** – случайная функция, прирост которой определяется генератором случайных чисел. Эта функция ведет себя чуть более предсказуемо, чем предыдущая.

  Дополнительные параметры:

  initValue(начальное_значение)

minValue(минимальное_значение)

maxValue(максимальное_значение)

minRate(минимальный_прирост)

maxRate(максимальный_прирост)

step(шаг)

Формулы:

$$f(0)=initValue$$

$$f(t+\Delta t)=clamp(f(t)+rate(t)\cdot\Delta t, minValue, maxValue)\ ,$$

$$rate(0)=random(minRate, maxRate)$$

$$rate(t+\Delta t)=rate(t) \quad \text{если} \quad \Delta t<step$$

$$rate(t+step)=random(minRate, maxRate)$$

Здесь $clamp(x, a, b)$ - функция, ограничивающая x до интервала $[a, b]$, т.е. $clamp(x, a, b)=min(max(x, a), b)$.

Прирост может быть и отрицательным (если минимальный прирост меньше нуля); в этом случае, прирост скорее можно было бы назвать убыванием.

Пример:

```
Function( MyFunction )
{
     type( RANDOM_RATE )
     initValue( 50 )
     minValue( 0 )
     maxValue( 80 )
     minRate( -1 )
     maxRate( 1 )
     step( 5 )
}
```

- **RANDOM_RATE2** – случайная функция, прирост прироста которой определяется генератором случайных чисел. Эта функция ведет себя еще чуть более предсказуемо, чем предыдущая.

Дополнительные параметры:

initValue(начальное_значение)

minValue(минимальное_значение)

maxValue(максимальное_значение)

initRate(начальный_прирост)

minRate(минимальный_прирост)

maxRate(максимальный_прирост)

minRate2(максимальный_прирост)

maxRate2(максимальный_прирост)

step(шаг)

bounce(отскок)

Формулы:

$$f(0)=initValue$$

$$f(t+\Delta t)=clamp(f(t)+rate(t)\cdot\Delta t, minValue, maxValue)\ ,$$

$$rate(0)=initRate$$

$$rate(t+\Delta t)=clamp(rate(t)+rate2(t)\cdot\Delta t, minRate, maxRate)$$

$$rate2(0)=random(minRate2, maxRate2)$$

$$rate2(t+\Delta t)=rate2(t) \quad \text{если} \quad \Delta t<step$$

$$rate2(t+step)=random(minRate2, maxRate2)$$

Причем, если значение функции достигает минимума или максимума, то rate умножается на bounce. Например, если bounce равен -1, то при достижении максимального или минимального значения rate поменяет знак на противоположный; а если bounce равен 0, то при достижении максимального или минимального значения rate обнулится, а если bounce равен 1, то ничего не произойдет.

Пример:

```
Function( MyFunction )
{
     type( RANDOM_RATE2 )
     initValue( 50 )
     initRate( 0.5 )
     minRate2( -0.1 )
     maxRate2( 0.1 )
     minRate( -1 )
     maxRate( 1 )
     minValue( 0 )
     maxValue( 80 )
}
```

- **RANDOM_MANUAL** – случайная функция, моделирующая некое явление, имеюще

набор стационарных состояний, с возможностью случайного перехода между соседними состояниями.

Дополнительные параметры:

initValue(начальное_значение)

minRate(минимальный_прирост)

maxRate(максимальный_прирост)

Состояния определяются так:

State(значение)

{

step(шаг_времени)

stayChance(шанс_остаться_в_том_же_состоянии)

upChance(шанс_перейти_вверх)

downChance(шанс_перейти_вниз)

}

Таких состояний должно быть не меньше двух. Функция работает так: через каждый шаг времени (step) используется генератор случайных чисел и шансы, указанные в определении состояния, чтобы определенить, что нужно делать: остаться в этом состоянии, перейти к состоянию с более высоким значением, или перейти к состоянию с более низким значением. Если выпадает переход, то в течении некоторого времени происходит линейное изменение значения функции с использованием maxRate (при переходе к большему значению), либо minRate (при переходе к меньшему значению).

Пример:

```
Function( MyFunction )
{
     type( RANDOM_MANUAL )
     initValue( 30 )
     minRate(-8)
     maxRate(8)
     State( 0 )
     {
          step( 100 )
          stayChance( 50 )
          upChance( 1 )
          downChance( 0 )
```

```
        }
        State( 10 )
        {
            step( 100 )
            stayChance( 30 )
            upChance( 2 )
            downChance( 5 )
        }
        State( 20 )
        {
            step( 70 )
            stayChance( 10 )
            upChance( 0 )
            downChance( 1 )
        }
    }
```