Kurz LSL skriptování

Shiny Iceberg

2009

součást cyklu
Nejsme jelita

Kurz LSL skriptování

Shiny Iceberg
v Second Life od roku 2006
shiny.iceberg@virtualmagazine.cz

Aktuální projekty

virtualmagazine.cz
Urbanica, Shinyland
Bwindi Orphans
cyklus Nejsme jelita

Organizační body

Průběh lekce

- bude trvat zhruba 90-120 minut
- pokud nekladete dotazy, vypněte si mikrofon
- příklady si klidně zkoušejte přímo v hledišti

Vaše otázky

- můžete se ptát na konci každého snímku
  nebo na konci celé přednášky
- dotazy mohou být přes voice nebo IM
- při psaní IM z posledních řad použijte Shout

Plán přednášky

1. Fungování a struktura skriptu
2. Z čeho se skládá skript
3. Vlastnosti objektu
4. Pohyb objektu
5. Pose bally
6. Komunikace skriptu
7. Inventory objektu
8. Detektory
9. Particles
10. Příklady a dokumentace

Detektor

Skupina funkcí *llDetected** slouží pro zjištění informací o objektech a avatarech ze

- snímacího senzoru
  event *sensor*
- kliknutí na objekt
  eventy *touch, touch_start, touch_end*
- kolize objektu
  eventy *collision, collision_start, collision_end*

Funkce pro detektory

- llSensor / llSensorRepeat
- llDetectedGroup
- llDetectedKey / llDetectedName
- llDetectedOwner
- llDetectedPos / llDetectedRot
- llDetectedType
- llDetectedTouchFace / llDetectedTouchUV
- llVolumeDetect

LSLwiki neobsahuje novější llDetected* funkce, je proto vhodné mrknout i na oficiální SL wiki stránky:

http://wiki.secondlife.com/wiki/Category:LSL_Detected

Eventy pro detektory

- sensor / no_sensor
- touch
- touch_start / touch_end
- collision
- collision_start / collision_end

```
llSensor(string Jmeno, key Id,
    integer Typ, float Dosah, float Uhel);
```

VSTUP:

- ***Jmeno:*** jméno hledaného objektu/avatara
- ***Id:*** UUID hledaného objektu/avatara
- ***Typ:*** typ hledaného objektu, může nabývat hodnot
  AGENT, PASSIVE, ACTIVE, SCRIPTED
- ***Dosah:*** dosah senzoru v metrech
- ***Uhel:*** směr snímání senzoru, hodnoty 0 až PI

VÝSTUP: nic

Poznámky k llSensor 1/2

- funkce nemá žádný výstup, ale dochází k aktivaci eventu *sensor* (pokud se něco našlo) nebo *no_sensor* (pokud detekce nic nenašla)
- detekce přes hranici regionu není spolehlivá
- maximální dosah detektoru je 96 metrů
- úhel snímání je prostorový úhel kolem osy x, hodnota PI znamená snímání ve všech směrech
- jméno a UUID hledaného objektu může být prázdné, pak se hledají všechny objekty
- detekuje se vždy střed objektů, nikoliv hranice

Poznámky k llSensor 2/2

- typ snímaného objektu je definován takto:
  - **AGENT:** avatar
  - **PASSIVE:** objekt bez běžícího skriptu, nebo se skriptem, který právě nic nevykonává, non-physical objekt anebo physical bez pohybu
  - **ACTIVE:** objekt s právě pracujícím skriptem nebo physical objekt v pohybu
  - **SCRIPTED:** objekt s právě pracujícím skriptem
- typy snímaného objektu lze kombinovat, např.

```
llSensor("", NULL_KEY, AGENT | ACTIVE, 25, PI)
```

```
llSensorRepeat(string Jmeno, key Id,
   integer Typ, float Dosah, float Uhel
   float Opakovani);
```

VSTUP:

- ***Jmeno:*** jméno hledaného objektu/avatara
- ***Id:*** UUID hledaného objektu/avatara
- ***Typ:*** typ hledaného objektu, může nabývat hodnot
  AGENT, PASSIVE, ACTIVE, SCRIPTED
- ***Dosah:*** dosah senzoru v metrech
- ***Uhel:*** směr snímání senzoru, hodnoty 0 až PI
- ***Opakovani:*** počet sekund mezi skenováním

VÝSTUP: nic

event *sensor* / *no_sensor*

```
sensor(integer Celkem)
{ ... }

no_sensor()
{ ... }
```

- do proměnné *Celkem* se uloží celkový počet detekovaných objektů/avatarů, maximálně ale 16 nejbližších

eventy *touch**

`touch_start(integer Celkem)`

- aktivuje se při stisknutí tlačítka myši

`touch(integer Celkem)`

- aktivuje se opakovaně při držení tlačitka myši

`touch_end(integer Celkem)`

- aktivuje se při uvolnění tlačítka myši

POZNÁMKA: do proměnné *Celkem* se uloží celkový počet avatarů, kteří kliknuli na objekt

Příklad k *llSensor*

```
default
{
    touch_start(integer Celkem) {
        llSensor("", NULL_KEY, AGENT, 10, PI);
    }

    sensor(integer celkem) {
        llWhisper(0, "Kolem je " + (string)celkem + " avataru" );
    }

    no_sensor() {
        llSay(0, "Zadny avatar neni nablizku");
    }
}
```

Funkce *llDetected**

Následující funkce lze použít pouze uvnitř eventů

- sensor
- no_sensor
- touch
- touch_start
- touch_end
- collision
- collision_start
- collision_end

```
integer llDetectedGroup(integer Poradi);
```

VSTUP: pořadové číslo zkoumaného objektu/avatara ze všech detekovaných, 0 je číslo nejbližšího avatara/objektu

VÝSTUP: vrací 0 (FALSE) nebo 1 (TRUE), což indikuje, zda má detektor a detekovaný objekt/avatar shodnou aktivní skupinu

```
key llDetectedKey(integer Poradi);          A
string llDetectedName(integer Poradi);      B
key llDetectedOwner(integer Poradi);        C
```

VSTUP: pořadové číslo zkoumaného objektu/avatara

VÝSTUP: vrací

- UUID detekovaného objektu/avatara **A**
- jméno detekovaného objektu/avatara **B**
- vlastníka detekovaného objektu **C**

POZNÁMKA:

- u objektů vlastněných skupinou vrátí **C** prázdný UUID (NULL_KEY)

```
vector llDetectedPos(integer Poradi);
rotation llDetectedRot(integer Poradi);
```

A
B

VSTUP: pořadové číslo zkoumaného objektu/avatara

VÝSTUP: vrací

- polohu detekovaného objektu/avatara (A)
- natočení detekovaného objektu/avatara (B)

```
integer llDetectedType(integer Poradi);
```

VSTUP: pořadové číslo zkoumaného objektu/avatara

VÝSTUP: typ detekovaného objektu/avatara

AGENT / PASSIVE / ACTIVE / SCRIPTED

POZNÁMKA: testování se provádí bitovým srovnáním

```
if (llDetectedType(0) & AGENT)
{
    // zde budou prikazy k provedeni
}
```

```
integer llDetectedTouchFace(integer Poradi);
```

VSTUP: pořadové číslo zkoumaného objektu/avatara

VÝSTUP: vrací číslo strany objektu, na který avatar kliknul myší

POZNÁMKA: tato funkce funguje pouze pro eventy touch*

Příklad k *llDetectedTouchFace*

```
default
{
    touch_start(integer Celkem)
    {
        integer Strana = llDetectedTouchFace(0);
        vector Barva = llGetColor(Strana);

        llSetColor(<1.0, 0.0, 0.0>, Strana);
        llSleep(0.5);
        llSetColor(Barva, Strana);
    }
}
```

```
vector llDetectedTouchUV(integer Poradi);
```

VSTUP: pořadové číslo zkoumaného objektu/avatara

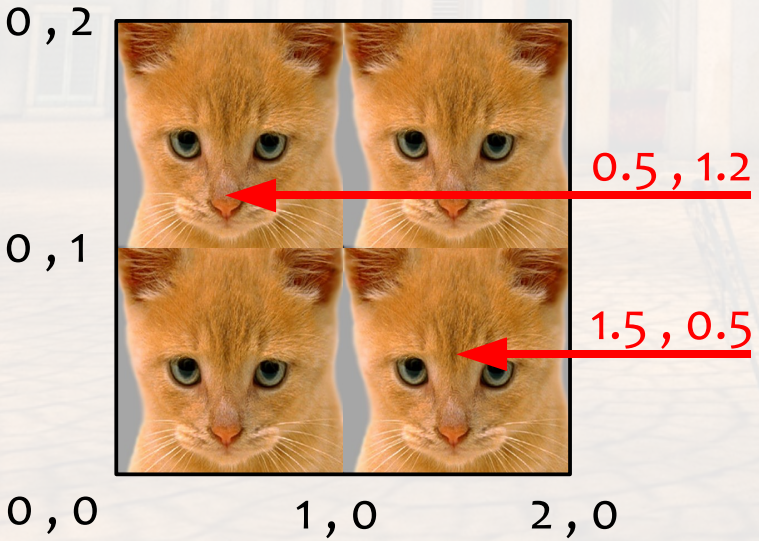
VÝSTUP: vrací souřadnice místa dotyku vztažené k textuře, na kterou bylo kliknuto

POZNÁMKA:

- tato funkce funguje pouze pro eventy touch*
- vrácený vektor
  - hodnoty X a Y odpovídají hledaným souřadnicím
  - hodnota Z je vždy 0.0
- existují obdobné funkce, např. *llDetectedTouchST*, která detekuje místo dotyku v souřadnicích strany

Poznámky k *llDetectedTouchUV*

```
default
{
    touch_start(integer Celkem)
    {
        vector Dotyk = llDetectedTouchUV(0);
        float U = Dotyk.x;
        float V = Dotyk.y;
        llOwnerSay("Misto dotyku je na souradnicich:");
        llOwnerSay( (string)U + " a " + (string)V );
    }
}
```



ukázka vlevo ukazuje, jak fungují souřadnice kliknutí u stran, kde je textura několikrát zopakovaná

eventy collision*

`collision_start(integer Celkem)`

- aktivuje se při začátku kolize objektu s jiným objektem nebo avatarem

`collision(integer Celkem)`

- aktivuje se opakovaně, dokud objekt koliduje

`collision_end(integer Celkem)`

- aktivuje se při konci kolize objektu

POZNÁMKA: kolize nastává i pokud se avatar postaví na objekt

```
llVolumeDetect(integer AnoNe);
```

VSTUP: TRUE/FALSE pro zapnutí phantom detekce

VÝSTUP: nic

POZNÁMKA:

- tato funkce při parametru TRUE změní objekt na phantom (nesmí být předem) a ačkoliv normálně nemůže nastat kolize, eventy *collision_start* a *collision_end* budou aktivovány v případě, že jiný objekt nebo avatar vstoupí do „objemu“ objektu
- event *collision* je v tomto případě nefunkční

Příklad: Uvítání návštěvníků

```
default
{
    state_entry()
    {
        llVolumeDetect(TRUE);
    }

    collision_start(integer Celkem)
    {
        key Avatar = llDetectedKey(0);
        llDialog(Avatar, "Chovejte se tady slusne", [], 1);
    }
}
```

Příklad: Auto otevírání dveří 1/3

```
// ===== Skript v detektoru =====
default
{
    on_rez(integer Start)
    {
        llResetScript();
    }

    state_entry()
    {
        string Vlastnik = "Dvere " + llKey2Name( llGetOwner() );
        llSetObjectName( Vlastnik );
        llVolumeDetect(TRUE);
    }

    collision_start(integer Celkem)
    {
        llSay(1, "otevrit");
    }
}
```

Příklad: Auto otevírání dveří 2/3

```
// ===== Skript ve dverich =====
dvere(integer Otevrit)
{
    vector RotovatRad;
    rotation Rotovat;

    if (Otevrit)
    {
        llSetPrimitiveParams([PRIM_PHANTOM, TRUE]);
        RotovatRad = <0, 0, -90 * DEG_TO_RAD>;
        Rotovat = llEuler2Rot(RotovatRad);
        llSetLocalRot( Rotovat * llGetLocalRot() );
    } else {
        RotovatRad = <0, 0, 90 * DEG_TO_RAD>;
        Rotovat = llEuler2Rot(RotovatRad);
        llSetLocalRot( Rotovat * llGetLocalRot() );
        llSetPrimitiveParams([PRIM_PHANTOM, FALSE]);
    }
}
```

Příklad: Auto otevírání dveří 3/3

```
default
{
    state_entry()
    {
        string Vlastnik = "Dvere " + llKey2Name( llGetOwner() );
        llListen(1, Vlastnik, NULL_KEY, "otevrit");
    }

    listen(integer Kanal, string Jmeno, key Id, string Text)
    {
        dvere(TRUE);
        state open;
    }
}

state open
{
    state_entry()
    {
        llSetTimerEvent(5.0);
    }

    timer()
    {
        llSetTimerEvent(0.0);
        dvere(FALSE);
        state default;
    }
}
```

Otázky a diskuze