O Unreal Script - [UE1] [UC] 2 Typy zmiennych by Raven dnia: 02 Grudzien 2007, 11:01
Jak kazdy jezyk programowania takze i Unreal Script posiada zmienne. Zmienne deklaruje sie nastepujaco:

Kod:
Var([nazwa]) [Prefixy] TypZmiennej NazwaZmiennej;

Typy
Istnieje kilka typów zmiennych:

byte
Liczba calkowita od 0 do 255

Unreal Script:
```
var byte MyByte;
 MyByte=255;
```

iInt
Liczba calkowita (32 bitowa)

Unreal Script:
```
var int MyInt;
 MyInt=11;
 MyInt++; //wartosc wieksza o 1
 MyInt--; //wartosc mniejsza o 1
```

float
Liczba zmiennoprzecinkowa

Unreal Script:
```
var float MyFloat;
 MyFloat=11.05;
 MyFloat*=2.;
 MyFloat = MyFloat/22;
```

bool
Zmienna logiczna (wartosc true lub false)

Unreal Script:

```
var bool MyBool;
 MyBool=false;
```

string
Ciag znaków ograniczony poprzez " i ":

Unreal Script:

```
var string MyString1;
 var string MyString2;
 var string MyString3;
 MyString1="ssdd1";
 MyString2="ddss2";
 MyString3= MyString1$MyString2;//laczy string
 iMyString3=Right(MyString1, 2); // zwraca 2 znaki z MyString2, zaczynajac od prawej strony
 MyString3=Left(MyString2, 2);// zwraca 2 znaki z MyString2, zaczynajac od lewej strony
 MyString3=Len(MyString1);// zwraca liczbe znaków w MyString1
```

name
Nazwa wlasciwie tez jest ciagiem znaków, z ta róznica, ze ograniczamy go znakami ' i '. Najlepiej jesli spojrzysz na nie jako na nazwy (tagi) przedmiotów w UT. Nie wykonuj na nich zadnych operacji, które wykonalbys na ciagach.

Unreal Script:

```
var name MyName;
 MyName='FirstName';
```

enum
Zmienna, która przyjmuje wartosc sposród kilku zdefiniowanych nazw

Unreal Script:

```
enum EMyEnum //struktura
 {
    MY_Enum1,
   MY_Enum2
 };
 var EMyEnum FirstEnumVar; // dostep poprzez FirstEnumVar
 FirstEnumVar=MY_Enum1;
```

struct

Struktura jest to zlozony typ danych, którego glównym zadaniem jest grupowanie zmiennych róznego typu. Skladnia wyglada nastepujaco:

Unreal Script:

```
struct nazwa [extends struktura_nadrzedna] {
   //tu deklaruje sie zmienne.
 };
```

Przyklad:

Unreal Script:

```
struct SMyStruct //struktura
 {
    var int IntInStruct;
    var float FloatInStruct;
    var string StringInStruct;
 };
 var SMyStruct FirstStructVar; // dostep poprzez FirstStructVar
 var SmyStruct FirstStructArray[3];
 (...)
 FirstStructVar.IntInStruct=1;
 FirstStructVar.FloatInStruct=1.05;
 FirstStructVar.StringInStruct="jeden"; /* Nadaje warosc JEDNEJ zmiennej z danej komórki
tabeli.*/
 (...)for(i=; i<3; i++)
 {
   FirstStructArray[i]. IntInStruct =1;
   FirstStructArray[i]. FloatInStruct =1.05;
   FirstStructArray[i]. StringInStruct ="jeden";
 }/* nadaje wartosci wszystkim zmiennym w FirstStructArray */
```

W defultproperties z kolei zmiennym w strukturze przypisuje sie wartosci tak

Unreal Script:

```
defaultproperties
 {
   FirstStructVar=(IntInStruct=1,FloatInStruct=1.05,SringInStruct ="jeden")
 }
```

Jedna z istniejacych struktur jest werkor.

Prefixy

const

Stala po nadaniu jej wartosci (domyslna wartosc zmiennej lub w UED) podczas dzialania skryptu nie mozesz jej zmienic.

Unreal Script:

```
var const bool MyBool;
```

editconst

Zmiennej nie mozna zmienic z poziomu UnrealED :]

Unreal Script:

```
var editconst bool MyBool;
```

input

Sprawia, ze mozemy taka zmienna zbindowac. Dziala tylko z typami byte i float

Unreal Script:

```
var input byte MyInputByte;
```

transient

Jest jedynie do uzycia tymczasowego. Przy ladowaniu obiektu jest zerowana.

Unreal Script:

```
var transient bool MyBool;
```

native

Zmienna jest czesciej uzywana przez C++ niz Unreal Script.

Unreal Script:

```
var native bool MyBool;
```

private

Zmienna bedzie prywatna, dostepna tylko dla skryptu

Unreal Script:

```
var private bool MyBool;
```

travel

Odnosi sie tylko do PlayerPawn oraz Inventory. Znaczy mniej wiecej tyle: "wartosc tej zmiennej musi podrózowac pomiedzy poziomami". Sprawia to, ze po zaladowaniu nowej mapy (tryb Single Player) bron, ilosc amunicji, stan zdrowia nie jest "zapominany"

Unreal Script:

```
var travel bool MyBool;
```

skip

Tylko do operacji logicznych (szczególy w licencji). Przykladu nie podam, bo nigdzie nie ma szczegółów :(

export

Tylko do klas Object i Actor (szczególy w licencji). Przykladu nie podam, bo nigdzie nie ma szczegółów :(

Tablice

Unreal Engine 1.x supportuje tylko jednowymiarowe tablice:

Unreal Script:

```
var int MyArInt[3];
```

Dostep do takiej zmiennej uzyskujemy nastepujaco:

Unreal Script:
```
JakasZmienna=MyArInt[i];
```

Kod:
```
Var([nazwa]) [Prefixy] TypZmiennej NazwaZmiennej;
```

Gdzie "i" oznacza liczbe z zakresu od 0 do wymiar_tablicy-1 czyli w naszym przypadku od 0 do 2.

Edycja z poziomu UnrealED

Uzyskujemy dzieki wstawieniu () po var. Deklaracja powinna wygladac nastepujaco:

Ilustruje to przyklad:

Unreal Script:
```
/*edytujemy ja z poziomu UED nazwa grupy=nazwa klasy */
 var() int MyInt;
 /*edytujemy ja z poziomu UED nazwa grupy=MyGroup=nazwa w nawiasie :]*/
 var(MyGroup) int MyInt2;
```

Jesli w () nie wpiszemy zadnej nazwy, to grupa takiej zmiennej bedzie nazwa klasy.

Struktury

Aby struktura mogla byc edytowana z poziomu UnrealED, odpowiednio zadeklarowane musza byc takze zmienne wewnatrz struktury:

Unreal Script:
```
struct nazwa [extends struktura_nadrzedna] {
   Var([nazwa]) [Prefixy] TypZmiennej NazwaZmiennej;
 };
```

Ilustruje to przyklad:

Unreal Script:

```
struct SMyStruct //struktura
{
    var() int IntInStruct;      //zmienna widoczna z poziomu UnrealED
    var() float FloatInStruct;  //zmienna widoczna z poziomu UnrealED
    var string StringInStruct;  //zmienna niewidoczna w UnrealED
};
var() SMyStruct FirstStructVar; // dostep poprzez FirstStructVar
```

Zmienne a Obiekty

Mozna takze zadeklarowac zmienne odnoszace sie do obiektów:

Unreal Script:

```
var actor A; // Odnosi sie do klasy Actor
var pawn P; // Odnosi sie do klasy Pawn
var texture T; // Odnosi sie do Textury
```

Nalezy pamietac, iz zmienna P moze przechowywac kazdy istniejacy w grze obiekt bedacy dzieckiem klasy Pawn.

Unreal Script:

```
var pawn P, S;
function MyFunction()
{
   if(P.Enemy == S)
      P.PlayRunning();
}
```

Zmienna P albo odnosi sie do jakiegos istniejacego w czasie dzialania skryptu Pawna, albo jej wartosc wynosi None. Dzieki temu mozemy ulepszyc powyzsza funkcje

Unreal Script:

```
var pawn P, S;
 function MyFunction()
 {
    if(P == none)  return; /* jesli pawn nie istnieje nie mamy nic do roboty i konczymy dzialanie funkcji, w przeciwnym wypadku kontynuujemy */
    if(P.Enemy == S) // jesli mamy przeciwnika I jest nim S
       P.PlayRunning(); // odtwarzamy aminacje biegania
 }
```

Oczywiscie mozna sie obyc bez warunku **if(P == none) return; i skrypt bedzie dzialac poprawnie. W momencie jednak, gdy któras ze zmiennych bedzie pusta (wartosc none) w logu zostanie zapisany slynny blad Accessed None. Kilka takich bledów nic nie zmieni, lecz duze ich nagromadzenie moze znaczaco spowolnic gre.**

Zmienne a Klasy

W powyrzszym przypadku, obiekty musialy istniec w grze, aby bylc uzyte. Zmiena jako klasa nie wymaga tego.

Unreal Script:

```
var() class C; // klasa do Spawnowania
 var actor A; // zmienna "przechowujaca" obiekt
 A = Spawn(C); // tworzy obiekt w grze
```

Zmienna C okresla klase, która ma zostac utworzona w grze (dowolna), zas A "przechowuje" obiekt. Dzieki niej mozemy sie odwolac do funkcji i zmiennych wewnatrz zespawnowanej klasy. Jest tylko jeden drobny szczegól... odwolac sie mozesz do wszystkiego zawartego w obiekcie ACTOR. Dlaczego? Bo odwolujesz sie do klasy Actor i nie mozesz "siegnac dalej". Aby uniknac ograniczenia, musimy zdefiniowac grupe:

Unreal Script:

```
var() class<nazwa_istniejacej_klasy> NazwaZmiennej;
```

Czy dzieki temu zapisowi mozemy sie odwolywac do wartosci innych niz te z klasy Actor? Niezupelnie, ale to jeden krok do przodu. Prosze zwrócic uwage, ze w UED bedziesz mógl wstawic tylko klase zawarta pomiedzy < i > lub klasy pochodne. No, ale przejdzmy dalej:

Unreal Script:

```
var() class< decoration > D; // klasa do Spawnowaniavar
 Decoration A; // zmienna "przechowujaca"
 A = Spawn(D); // tworzy obiekt w grze
```

Dopiero po zdefiniowani kolejnej zmiennej odnoszacej sie do obiektu (w tym przypadku Decoration) mozemy zespawnowac odpowiednia klase i odnosic sie do funkcji i zmiennych w niej zawartych. W tym przypadku musimy takze wpisac w **defaultproperties odnosnik do klasy.**

Unreal Script:

```
defaultproperties
{
    D=class'Botpack.Bin2'
}
```

Istniejace struktury

Vector

Punkt w przestrzeni 3D. Opisany za pomoca wartosci na osiach X, Y oraz Z

Unreal Script:

```
struct Vector
{
      var() config float X, Y, Z;
};
```

Plane

Punkt w 3D wraz z punktem odniesienia od srodka czyli X, Y, Z oraz W.

Unreal Script:

```
struct Plane extends Vector
{
      var() config float W;
};
```

Rotator

Rotacja (obrót) jest opisana za pomoca Pitch, Yaw, i Rol.

Unreal Script:

```
struct Rotator
{
	var() config int Pitch, Yaw, Roll;
};
```

Color

Kolor RGB. Jesli nie wiesz co to takiego, poszukaj w wikipedi lub pogoogluj.

Unreal Script:

```
struct Color
{
	var() config byte R, G, B, A;
};
```

Coords

Definicja plaszczyzny w 3D

Unreal Script:

```
struct Coords
{
	var() config vector Origin, XAxis, YAxis, ZAxis;
};
```

Region

Region w którym znajduje sie dany actor

Unreal Script:

```
struct PointRegion
{
        var zoneinfo Zone;       // Zone.
        var int      iLeaf;      // Bsp leaf.
        var byte     ZoneNumber; // Zone number.
};
var const PointRegion     Region;
```

Kod:

```
local [TypZmiennej] [NazwaZmiennej];
```

Zmienne lokalne i globalne

Do tej pory pisalem o zmiennych globalnych, czym jednak sa zmienne lokalne? W praktyce nie róznia sie one za bardzo od zmiennych globalnych, z tym wyjatkiem, ze do zmiennych globalnych maja dostep wszystkie funkcje w skrypcie, do nich mozna sie takze odwolywac poprzez inne skrypty, oraz ustawiac wartosci z poziomu UnrealEd. Zmienne lokalne zas dostepne sa tylko dla funkcji w której zostaly zadeklarowane i, w przeciwienstwie do zmiennych globalnych, moga sie powtarzac (oczywiscie nie w jednej funkcji i nazwa nie moze byc taka sama jak zmienna globalna). Zmienne lokalne deklaruje sie w nastepujacy sposób:

Skad jednak podzial na zmienne lokalne i globalne i jaki sens jest uzywania tych pierwszych? Otóz zmienne lokalne sa stosowane w funkcjach, w momencie, gdy maja sluzyc do wykonania pewnych obliczen, których wynik nie trzeba na dluzej zapamietywac. Przesledzmy to na nastepujacej funkcji:

Unreal Script:

```
function Draw100Lines()
{
    local int i;    for(i=; i<100; i++;)
   {
        BroadCastMessage("Liczymy do 100. Jest to liczba: "$i+1);
    }
}
```

Ta prosta funkcja wypisuje liczby od 0 do 100. Nie chodzi jednak o funkcje a o zmienna i. Teoretycznie móglbym zadeklarowac ja jako globalna, ale w praktyce takie postepowanie nie mialoby najmniejszego sensu. Przyjrzyjmy sie jednak takiej funkcji:

Unreal Script:

```
function PowiekszI()
{
    local int i;
    i=i+1;
}
```

Najciekawsze w tej funkcji jest to, ze chocbysmy wywolywali ja 1000 razy, maksymalna wartoscia jaka osiagnie zmienna i bedzie 1. Dzieje sie tak dlatego, gdyz podczas kazdego wywolania funkcja powieksza i o 1, zas kolejne wywolania nie maja dostepu do poprzedniej wartosci. Jesli chcielibysmy aby wartosc zmiennej i byla zapamietywana, funkcja powinna miec taki wyglad:

Unreal Script:

```
var int i;
function PowiekszI()
{
    i=i+1;
}
```