

Unreal Hacking

Christian Esperer
esperer@sec.informatik.tu-darmstadt.de
DECT: 5230

September 16, 2007

ToC

- ① Introduction to Unreal
- ② UnrealScript Coding
- ③ Modding
- ④ Deus Ex

Part I

Introduction to Unreal

Some facts about the Unreal Engine

- Originally used as 3D first-person ego-shooter game engine
- Functionality similar to that of engines like Quakell
- Created by Tim Sweeney, first release in 1998
- Written in C++ when using C was still common
- Emphasis on functionality and elegance, not speed
- Unreall uses outdated cylindric collision detection

Powered by Unreal. . .

- Unreal + UnrealTournament
- Star Trek: Deep Space Nine
- America's Army 1+2
- Rainbow Six
- Thief: Deadly Shadows
- Splinter Cell
- Star Wars Republic Commando
- Deus Ex

Concepts of UnrealScript

- Keep the algorithms fast → C++
- Keep the logic clean and simple → US
- Basic Unreal features are reflected in US
 - Latent functions
 - Replication
 - State blocks
 - Transparent serialization
 - Transparent multithreading

A first look at UnrealScript

UnrealScript is the core of the Unreal engine. It. . .

- was created from scratch by Tim Sweeney for Unreal I
- gets compiled to bytecode (like java, .net)
- runs on a VM (again, like java. . .)
- doesn't let you create threads, but creates them automatically where necessary
- has a garbage-collector
- runs in a sandbox
- has pointers, but no pointer arithmetic
- is platform-independent
- is approximately 20-50 times slower than C++

Power of UnrealScript

Huge parts of the base system are written in US:

- The bot AI code
- Most of the inventory handling
- The complete GUI
- Weapon functionality
- Keyboard functions like *select weapon*
- Stats webserver

Differences to Java

Java has it, UnrealScript doesn't. . .

- A debugger
- *Explicit* support for threads
- Explicit access to mutexes or semaphores
- Means to access the file system directly

That stuff isn't needed though, 'cause

- Complicated code goes in native libraries
- Every class runs in its own thread
- Synchronization is done in native code
- Serialization is handled by the VM

Noteworthy facts of UnrealScript

Different style. . .

- Members and methods are generally declared public
- Variables are partly prefixed
- Operators can be overloaded
- Basic functions are native static functions of the Object class
- US packages contain *both* byte- and sourcecode
- Java has no goto – UnrealScript requires it

Integration into the engine

UnrealScript integrates neatly into the rest of the engine

- Every in-game object has its UnrealScript class
- The complete VM state can be serialized
- A Server-Client protocol (multiplayer!) is integrated into the language
- All subsystems are accessible through UnrealScript
- Subsystems report their states to UnrealScript
- State code makes huge switch blocks and explicit threads obsolete

Extension made easy

- Unreal is split in modules
- Modules can be exchanged independently
- Compiler creates a .hpp for each UnrealScript class on request
- Native code has full access to UnrealScript variables
- Native code can call UnrealScript events
- Native code libraries get loaded on demand

Part II

UnrealScript coding

Parts of an UnrealScript file

An UnrealScript file consists of seven parts

- 1 Formal declaration
- 2 Variable declaration
- 3 Replication section
- 4 Native function declaration
- 5 Method implementation
- 6 State blocks
- 7 Default properties

Formal declaration

Like in java, each class must be named after its filename (or the other way round).

Each file begins with

class MyClass extends Object;

Important class modifiers:

- native
- nativereplication
- abstract
- config (section)
- guid(a, b, c, d) (reserved)

Variable declaration

Class variables, enums, structs all go here

- Members usually public
- Declaration modifiers available
- Syntax: `var [[[CATEGORY]]] MODIFIER TYPE varName [];`
 - (global)config
 - (edit)const
 - transient
 - native (concerns serialization only)
 - travel
 - localized
 - public, protected, private
- Brackets immediately after var make variable visible in editor property window
- A category can be specified in the brackets

Name vs. String

Strings...

- Contain arbitrary data
- Are mutable
- Can be localized
- Comparison is expensive

Names...

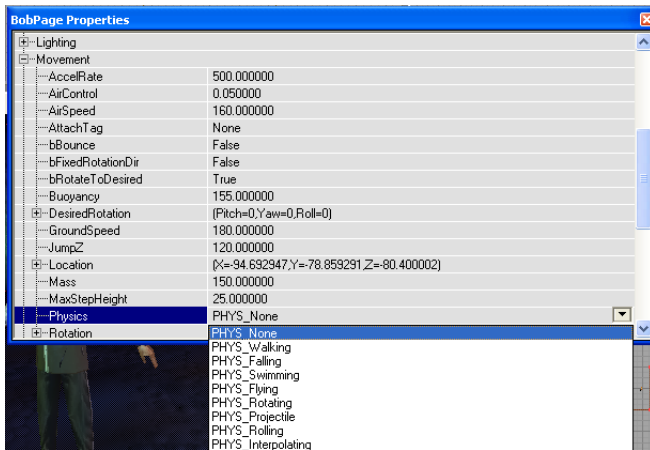
- Must match $[a-zA-Z][a-zA-Z0-9]^*$
- Used to map strings to IDs
- Case-insensitive
- Immutable
- Limited global pool of names
- Used for variables, classes, textures, sounds, ...
- Comparison is fast

Strings cannot easily be converted to names

Declaration Example

```
class Actor extends Object
    abstract
    native
    nativereplication;
var(Advanced) const bool bStatic;
var(Advanced) const bool bNoDelete;
var bool bAnimByOwner; // Animation dictated by owner.
var(Movement) const enum EPhysics {
    PHYS_None,
    PHYS_Walking,
    PHYS_Falling,
    PHYS_Rotating } Physics;
var ENetRole Role;
var(Networking) ENetRole RemoteRole;
```

Declaration Example



Replication

- Synchronization for multiplayer games
- Intention: save bandwidth
- All *Actor*-derived classes can be replicated
- Different code parts are executed on the server/the client
- Server (usually) is authoritative
- Client can “simulate” code for smoother appearance (Velocity)
- Replication of variables is asynchronous → very scalable

Replication

- Reliable/unreliable replication possible
- Variables are replicated asynchronously
- Functions can be used for RPC
- Calls possible in one direction per function
- *Simulated* functions are executed both by the server and the client

Replication roles:

- ROLE_Authority
- ROLE_AutonomousProxy
- ROLE_SimulatedProxy
- ROLE_DumbProxy
- ROLE_None

Replication – example

```
replication {  
    // client to server  
    reliable if (Role < ROLE_Authority)  
        AugmentationSystem, SkillSystem, BarkManager, FrobTarget,  
        FrobTime, inHand,...;  
    // server to client  
    unreliable if (Role == ROLE_Authority)  
        Location, Rotation;  
    // Functions the client can call  
    reliable if (Role < ROLE_Authority) 10  
        DoFrob, ParseLeftClick, ParseRightClick, ReloadWeapon, ActivateBelt;  
    // Functions the server can call  
    reliable if (Role == ROLE_Authority)  
        ClientMessage;  
}
```

Extension made easy – Native Functions

- Native functions are implemented in a system library
- C++ only officialy supported language
- Static native functions have an integer UID
- Use native code only
 - for cpu-intensive stuff
 - for security-critical (“suid”) stuff
 - for platform-dependent stuff
- Example: <http://deusex.hcesperer.org/tools/hcsqlib01.tar.bz2>

Native class implementation – UnrealScript part

```
class SQLITEFile expands Actor
    native;
var bool bLogQueries;
var string sLastError;
native function bool Open(String s);
native function int Query(String s);
native function int FetchRow(out String sCol0,
    out String sCol1, out String sCol2,
    out String sCol3);
native function Close();
function string Escape(string s){;}
//...
```

10

Native class implementation – native part

```
class HCSQLIB_API ASQLITEFile : public AActor {  
public:  
    BITFIELD bLogQueries:1 GCC_PACK(4);  
    FStringNoInit sLastError GCC_PACK(4);  
    DECLARE_FUNCTION(execClose);  
    DECLARE_FUNCTION(execFetchRow);  
    DECLARE_FUNCTION(execQuery);  
    DECLARE_FUNCTION(execOpen);  
    DECLARE_CLASS(ASQLITEFile,AActor,0);  
    ASQLITEFile();  
protected:  
    sqlite3* sqlfile; sqlite3_stmt* stmt;  
};
```

10

Method declaration

- Methods implement the main functionality of an US class
- Events can be called from native code
- Prefix *exec* makes them callable from the game console (works only in certain classes)
- Simulated functions run both client- and serverside
- Methods can be declared *singular* to prevent re-entry
- Usage example: Declare the *bump* event function singular if you want to move its actor inside it
- Methods can be overridden in child classes

State code

- Each class can define one or more states
- Only one state can be active at a time
- Usually used for AI programming, but use is not limited to that
- Each state can declare methods
- States can be derived
- Functions in a state override class-global methods
- Each state has a stackless code part
- Only stackless code can execute latent functions

State code

- Changing state via *GotoState*
- Inside a state, several labeled *blocks* exist
- Jump to the head of a block via *goto*
- No conditional jumps
- No calls/returns
- Engine can save state code “instruction pointer”

State code – example

```
state flying{
Begin:
    PlayFlying();
StartFlying:
    PickInitialDestination();
    MoveTo(destLoc);
Fly:
    if (ReadyToLand()) Goto('Land');
    PickDestination();
KeepGoing:
    CheckStuck();
    MoveTo(destLoc);
    Goto('Fly');
Land:
    if (!PickFinalDestination()) {
        PickDestination(); foobar();
    }
}
```

10

Object – the root of all classes

- Abstract base class
- Each object has a *name*, and a *class*
- Important structs like *Vector*, *Rotator*... are defined in Object
- Basic operators like $+$, $-$, $/$... are defined as native functions
- Basic functions are defined in Object
 - Math functions, String functions, class handling functions

Some unreal classes

Object

- Actor
- Bitmap
 - Texture
 - FractalTexture
 - ScriptedTexture
- Canvas
- CommandLet
- Console
- Subsystem
 - AudioSubsystem
 - Engine
 - Input
 - NetDriver
 - RenderBase

Everything in the world is an *Actor*

The class *Actor* is special:

- Base class of all in-game objects
- Each actor can physically interact with the world
- Special *Spawn* method to instantiate
 - Example: Place a soldier 50 worldunits in front of us
 - `mySoldier = Spawn(class'Soldier',,, Location + Vector(Rotation) * 50, Rotation);`
- *tick*-event for all non-static actors
- Events like *HitWall*, *Falling*, ...

Actors in Unreal

- Each actor has a physical definition
- Location, Rotation, collisionHeight, collisionRadius
- Physics
 - None
 - Falling
 - Rotating
 - Flying
 - Interpolating
- Actors can be replicated in multiplayer games
- Actors can serve as a light source
- Actors can serve as a sound source
- Each actor can define its in-game appearance

Some important actor properties

- bNoDelete, bStasis, lifeSpan
- bHidden, bHiddenEd, bMovable
- bBlock(Actors|Players), bCollide(Actors|World)
- collisionHeight, collisionRadius
- Location, Rotation, Physics, Velocity
- Mesh, Skins, DrawScale, Style
- Tag, Event
- initialState
- NetPriority, NetUpdateFrequency, bNetInitial, bNetOwner, Role, RemoteRole. . .
- AmbientSound, SoundRadius, SoundVolume, SoundPitch

Some Actor subclasses

- Brush
 - Mover
- Decoration
- Effects
- Info
- Inventory
- Keypoint
- Light
- NavigationPoint
- Pawn
- Projectile
- Triggers

Vectors and Rotators in Unreal

```
// A point or direction vector in 3d space.  
struct Vector {  
    var() config float X, Y, Z;  
};
```

```
// An orthogonal rotation in 3d space.  
struct Rotator {  
    var() config int Pitch, Yaw, Roll;  
};
```

Vectors and Rotators in Unreal

$$\vec{X} = \begin{pmatrix} 1 \\ 2 \\ 3 \end{pmatrix} \quad (1)$$

$$y = |\vec{X}| \quad (2)$$

$$dist = |\vec{E} - \vec{P}| \quad (3)$$

```
var vector X, E, P;
var float y, dist;
```

```
X = vect(1, 2, 3);
y = vsize(X);
```

```
dist = ~0;
if (Enemy != null) {
    E = Enemy.Location;
    P = Location;
    dist = VSize(E - P);
}
```

10

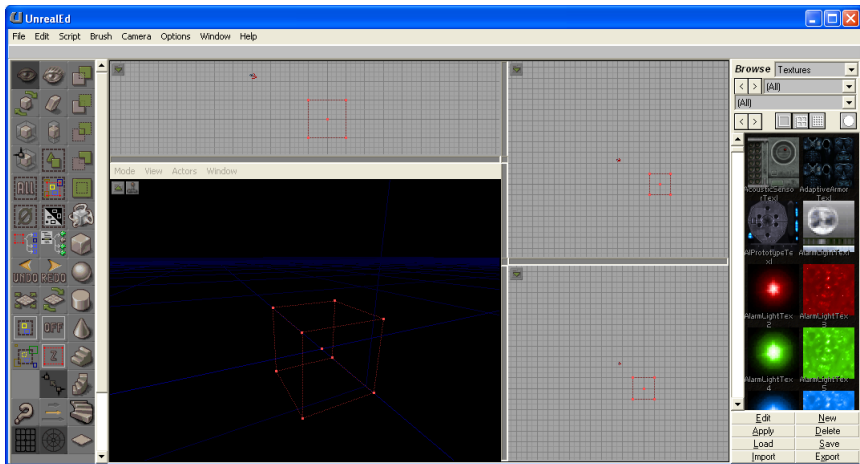
Part III

Modding

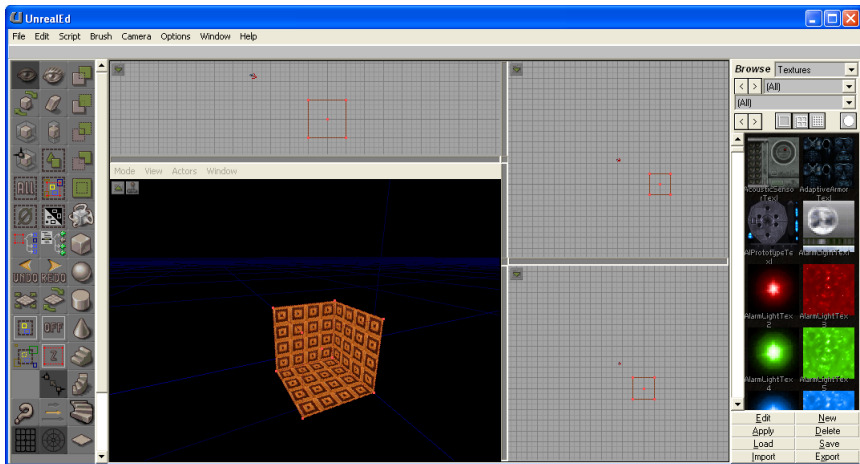
Mapping basics – the first room

- A new map is filled with *dark matter*
- First step: cut some part out
 - Use the cube builder (Editor.CubeBuilder) to create a cubical brush
 - Select a neat texture
 - Substract the brush from the world
 - Put some torches inside to light it up
 - Render the result

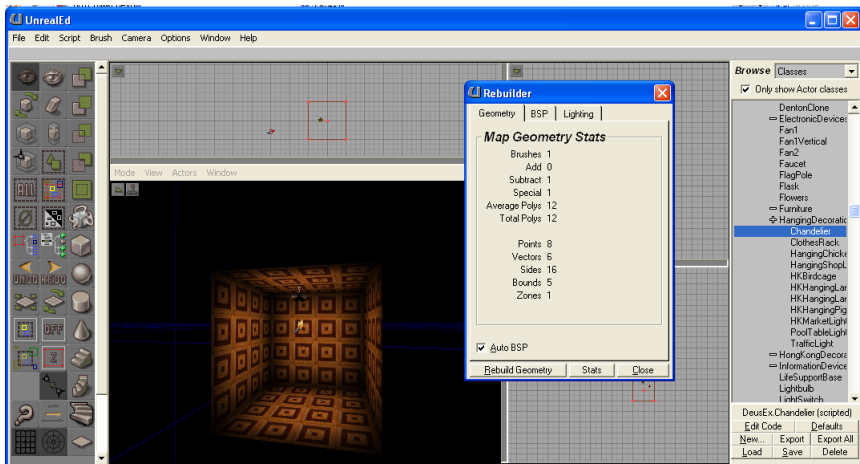
A new Map – Build the Brush



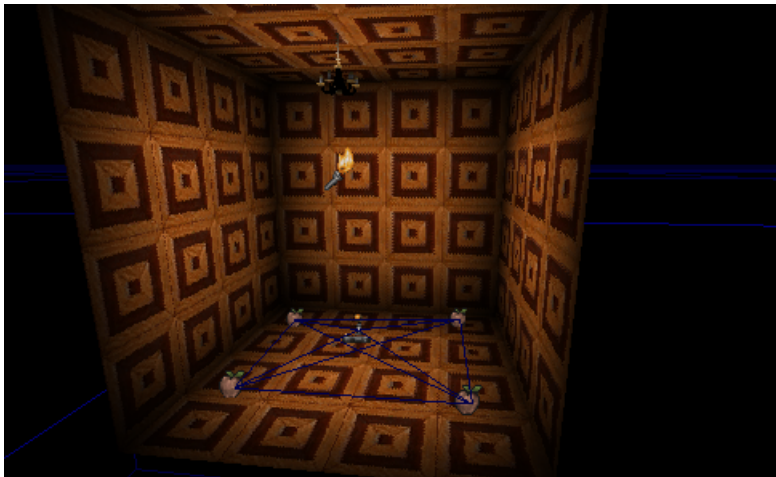
A new Map – Subtract Brush from the World



A new Map – Place Light(s) and Render



A new Map – Place Pathnodes and PlayerStart



Part IV

Deus Ex

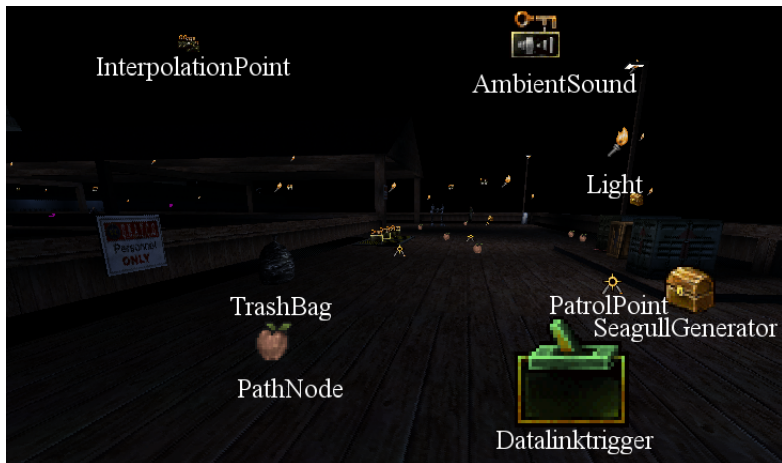
Deus Ex Additions to Unreal

- Alliance System
- Conversation System
- Extension to Weapons+Inventory System
- In-game Texts (Datacubes and Books)
- Mission/Flag System
- Object Interaction System
- Situation-Based Music

Opening scene in Deus Ex



Opening scene in Deus Ex



Triggers

- Used to perform some in-game action
- Action can be triggered by *touching* or *triggering*
- Each object has a *tag*
- Trigger's *event* property specifies which objects to work on
- A trigger has a tag, too → can be triggered by other triggers
- Arbitrary objects can serve as triggers
- In Deus Ex, a door can trigger some event if it finished opening/closing Example: A brick wall in the Liberty Island prison cell opens the prison door if moved

Triggers in Deus Ex

Some of those are available in the core Unreal Engine, too

- AllianceTrigger
- ConversationTrigger
- DataLinkTrigger
- FlagTrigger
- GoalCompleteTrigger
- InterpolateTrigger
- OrdersTrigger
- ShakeTrigger
- SkillAwardTrigger

NavigationPoints

Navigation points are used to mark navigation specific points
Some of these are Deus Ex specific, others exist in the core Unreal Engine, too

- AmbushPoint
- HidePoint
- LiftCenter, LiftExit
- MapExit
- **PathNode**
- PlayerStart
- SpawnPoint
- Teleporter

KeyPoints

Keypoints are used to mark things in the game

- AmbientSound
- AmbientSoundTriggered
- Block(All|Monsters|Players)
- CameraPoint
- InterpolationPoint

Flags in Deus Ex – handling the game logic

- Flags serve as per-game global variables
- Used to store several game states
- Examples: PaulDentonMeet_Played, M01PlayerAggressive, TerroristCommander_Dead
- Each flag can expire at the end of a mission
- Flags are stored to disk per savegame

Deus Ex missions

- Deus ex is divided in missions 0-15
- Missions are logical parts of the game
- Player can travel around the maps of one *mission*
- Several mission objectives exist per mission
- Mission scripts are used to implement more complex in-game logic
- Flags can expire at the end of a mission

Mission script excerpt

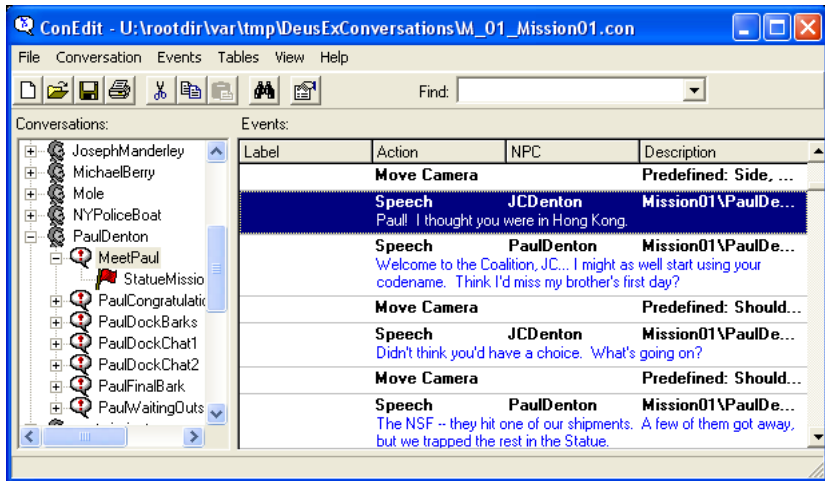
```
if (localURL == "01_NYC_UNATCOISLAND") {  
    if (!flags.GetBool('M01PlayerAggressive')) {  
        count = 0;  
        // count the living  
        foreach AllActors(class'Terrorist', T) count++;  
        // add the unconscious ones to the not dead count  
        // there are 28 terrorists total on the island  
        foreach AllActors(class'TerroristCarcass', carc) {  
            if ((carc.KillerBindName == "JCDenton") &&  
                (carc.itemName == "Unconscious"))  
                count++;  
            else if (carc.KillerBindName != "JCDenton") count++; }  
        // if the player killed more than 5, set the flag  
        if (count < 23)  
            // don't expire until mission 6  
            flags.SetBool('M01PlayerAggressive', True, , 6); }}
```

10

Introduction to the Conversation System

- Used for in-game Conversations
- Each *Actor* has a *BindName*
- → Any Actor can be a conversation partner
- Each conversation is a list of Commands
 - Speech
 - Choice
 - Move Camera
 - Play Animation
 - (Conditional)Jump
 - Transfer Object
 - Trigger something

Mission 1 – First Convo



ConEdit - U:\rootdir\var\tmp\DeusExConversations\01_Mission01.con

File Conversation Events Tables View Help

Find:

Conversations:

- JosephManderley
- MichaelBerry
- Mole
- NYPoliceBoat
- PaulDenton
 - MeetPaul
 - StatueMissio
 - PaulCongratulatic
 - PaulDockBarks
 - PaulDockChat1
 - PaulDockChat2
 - PaulFinalBark
 - PaulWaitingOuts

Events:

Label	Action	NPC	Description
	Move Camera		Predefined: Side, ...
Speech		JCDenton	Mission01\PaulDe... Paul! I thought you were in Hong Kong.
Speech		PaulDenton	Mission01\PaulDe... Welcome to the Coalition, JC... I might as well start using your codename. Think I'd miss my brother's first day?
	Move Camera		Predefined: Should...
Speech		JCDenton	Mission01\PaulDe... Didn't think you'd have a choice. What's going on?
	Move Camera		Predefined: Should...
Speech		PaulDenton	Mission01\PaulDe... The NSF -- they hit one of our shipments. A few of them got away, but we trapped the rest in the Statue.

References

- Unreal + Deus Ex UnrealScript Sourcecode
- http://wiki.beyondunreal.com/wiki/UnrealScript_Language_Reference
- <http://unreal.epicgames.com> (not available anymore)
- #dxediting on starchat