Game Engine Architecture IT3935

Lecture 13

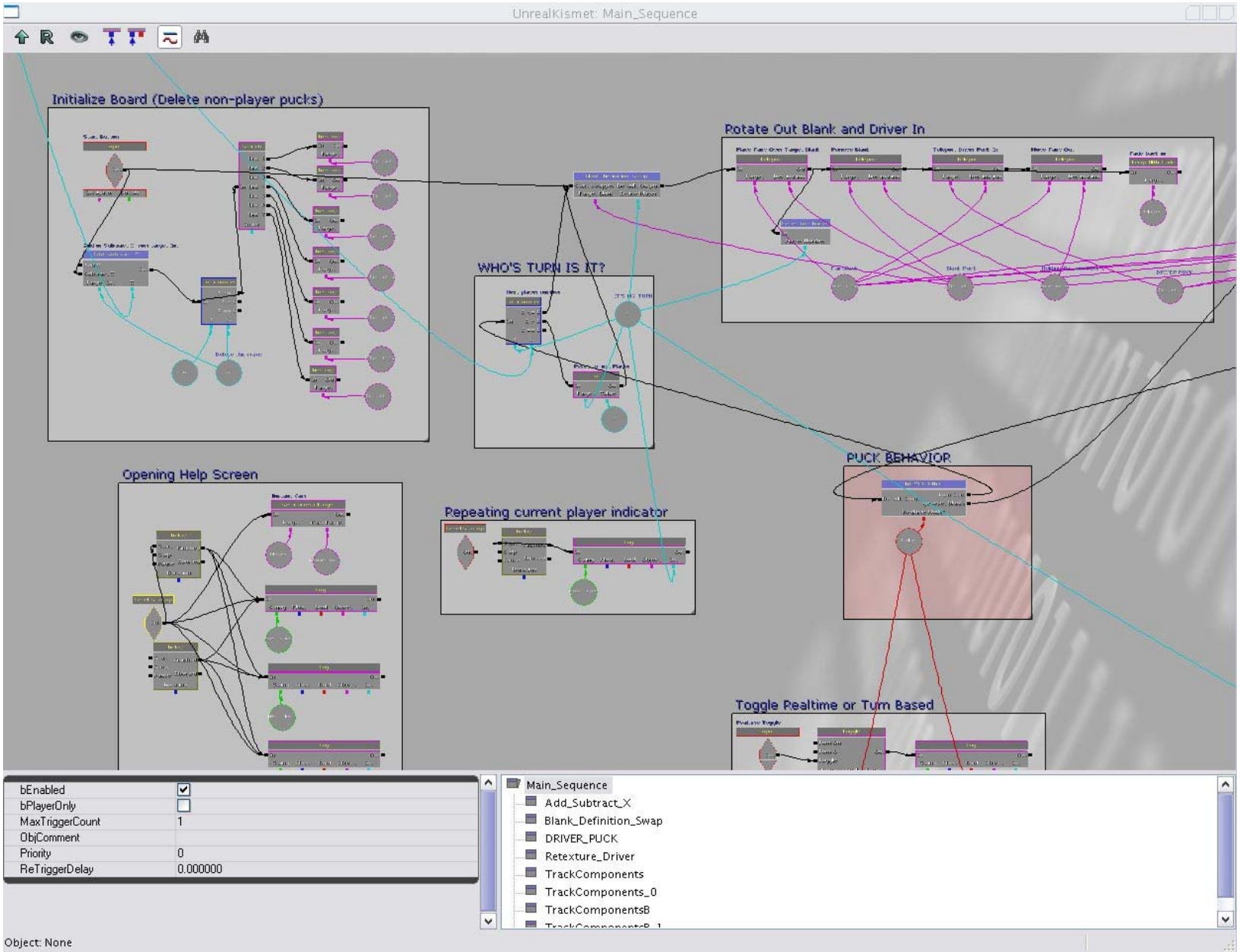
Summary

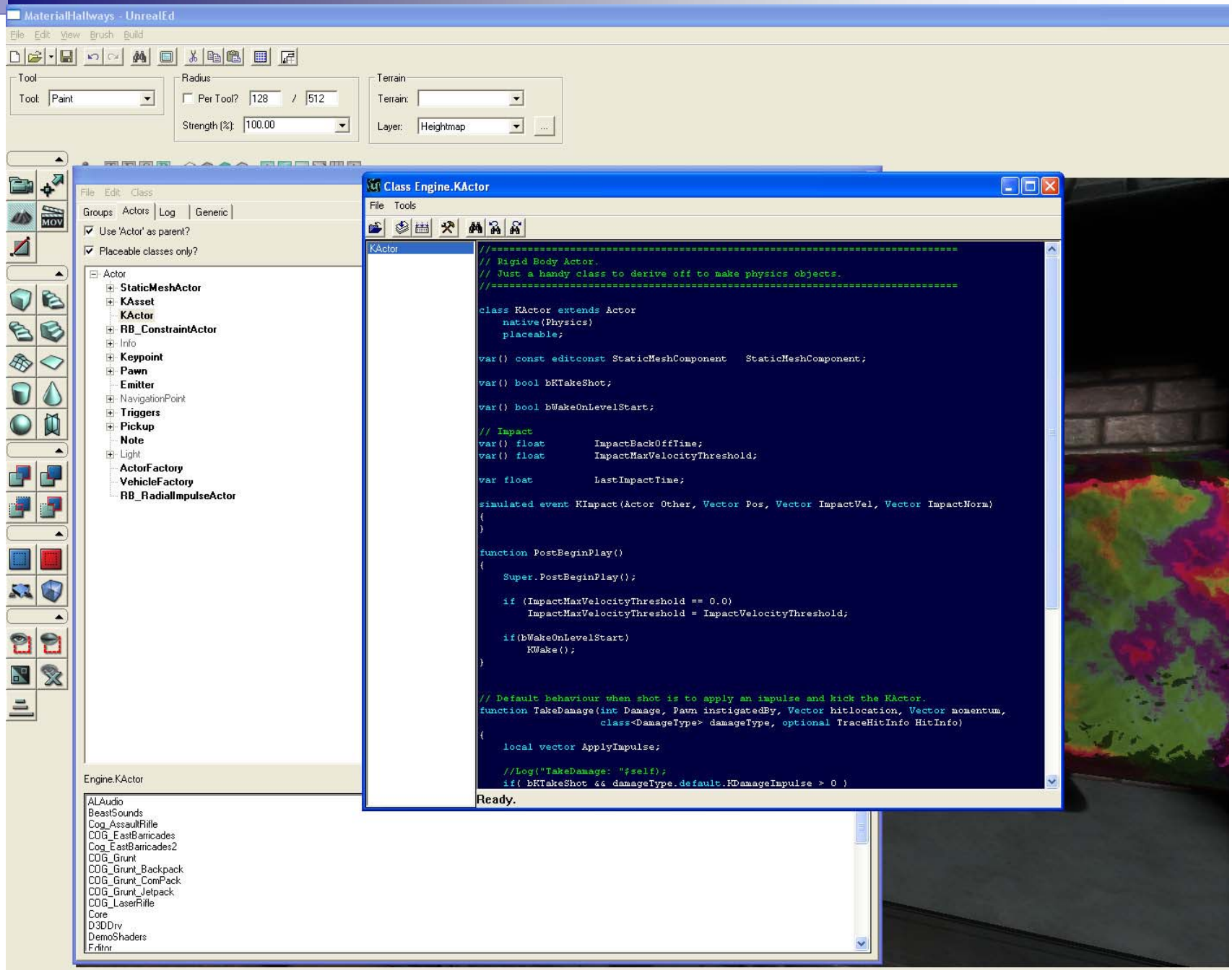
- Scripting Languages
- AI Libraries
- Engine Evaluation

Scripting Languages

- Scripting languages allows customizability by separating data and code and interpreted (or compiled) at runtime
- Allow designers to customize behavior without compiling code
- Often scripts can be text based (precompiled binary or text) or UI based

Visual UI Script Editor




MaterialHallways - UnrealEd
File Edit View Brush Build
Tool
Tool: Paint
Radius
Per Tool? 128 / 512
Strength (%): 100.00
Terrain
Terrain:
Layer: Heightmap
File Edit Class
Groups Actors Log Generic
Use 'Actor' as parent?
Placeable classes only?
Actor
StaticMeshActor
KAsset
KActor
RB_ConstraintActor
Info
Keypoint
Pawn
Emitter
NavigationPoint
Triggers
Pickup
Note
Light
ActorFactory
VehicleFactory
RB_RadialImpulseActor
Engine.KActor
ALAudio
BeastSounds
Cog_AssaultRifle
COG_EastBarricades
Cog_EastBarricades2
COG_Grunt
COG_Grunt_Backpack
COG_Grunt_ComPack
COG_Grunt_Jetpack
COG_LaserRifle
Core
D3DDrv
DemoShaders
Editor
Class Engine.KActor
File Tools
KActor
//==
// Rigid Body Actor.
// Just a handy class to derive off to make physics objects.
//==
class KActor extends Actor
native(Physics)
placeable;
var() const editconst StaticMeshComponent StaticMeshComponent;
var() bool bKTakeShot;
var() bool bWakeOnLevelStart;
// Impact
var() float ImpactBackOffTime;
var() float ImpactMaxVelocityThreshold;
var float LastImpactTime;
simulated event KImpact(Actor Other, Vector Pos, Vector ImpactVel, Vector ImpactNorm)
{
}
function PostBeginPlay()
{
Super.PostBeginPlay();
if (ImpactMaxVelocityThreshold == 0.0)
ImpactMaxVelocityThreshold = ImpactVelocityThreshold;
if(bWakeOnLevelStart)
KWake();
}
// Default behaviour when shot is to apply an impulse and kick the KActor.
function TakeDamage(int Damage, Pawn instigatedBy, Vector hitlocation, Vector momentum,
class<DamageType> damageType, optional TraceHitInfo HitInfo)
{
local vector ApplyImpulse;
//Log("TakeDamage: "$self);
if(bKTakeShot && damageType.default.KDamageImpulse > 0)
Ready.

Common Scripting Scenarios

- Loading resource (meshes, audio, animation etc)
- Setting position and orientation (Game units, GUI etc)
- Control in game behavior (lighting, volume, debug display, in game console)
- Customize specific object behavior (AI, effect)

Common scripting options

- Using an ini file to store settings (SDLConfig)
- Command line arguments (argv, argc)
- External scripts (text or binary) running through an embedded scripting engine.

Scripting Languages

- Programming language scripting
  - C#, Java,Perl, PHP, Python etc.
- Embedded scripting languages require a “host” to run the scripting in
  - Angelscript, GameMonkey, Javascript, Lua, Squirrel etc.
- Custom scripting system specific to application
  - *QuakeC, MEL, Nwscript, Unrealscript etc.*

Scripting Language problems

- Scripting (interpreted) is inherently much slower than compiled assembly code
- Scripting languages are usually not statically compiled, so errors are discovered at runtime, not compile time like C++.

AI

I always lie

AI in Games

- Decision and Action Response

FSM, Decision Tree

- Terrain Navigation

A Star , Breadth First Search*

- Behavior Learning

Neural net, Fuzzy Logic

Available Libraries

- OpenSource

- OpenSteer, MetaAgent, MicroPanther

- Path Finding

- PathEngine

- Commercial AI Engines

- AI Implant, Kynapse, Simbionic

Game Engine Evaluation

- http://www.devmaster.net

End

- Questions?