1. Record Nr. UNINA9910814436103321

Autore	Lundgren Filip
Titolo	CryENGINE game programming with C++, C#, and Lua / / Filip Lundgren, Ruan Pearce-Authers
Pubbl/distr/stampa	Birmingham : , : Packt Publishing, , 2013
ISBN	1-84969-591-1
Edizione	[1st ed.]
Descrizione fisica	1 online resource (276 pages) : illustrations
Collana	Community experience distilled
Altri autori (Persone)	Pearce-AuthersRuan
Disciplina	005.384
Soggetti	Video games - Programming Video games - Design
Lingua di pubblicazione	Inglese
Formato	Materiale a stampa
Livello bibliografico	Monografia
Note generali	Includes index.
Nota di contenuto	Intro -- CryENGINE Game Programming with C++, C#, and Lua -- Table of Contents -- CryENGINE Game Programming with C++, C#, and Lua -- Credits -- About the Authors -- About the Reviewers -- www.PacktPub.com -- Support files, eBooks, discount offers and more -- Why Subscribe? -- Free Access for Packt account holders -- Preface -- What this book covers -- What you need for this book -- Who this book is for -- Conventions -- Reader feedback -- Customer support -- Errata -- Piracy -- Questions -- 1. Introduction and Setup -- Installing Visual Studio Express 2012 -- Choosing your CryENGINE installation type -- Downloading the book's CryENGINE sample installation -- What just happened? -- Using a custom or newer CryENGINE installation -- Verifying that the build is functional -- Integrating CryMono (C# support) -- Compiling the CryMono project -- Loading and initializing CryMono via the CryGame.dll library -- Including the CryMono interface folder -- Initializing CryMono at start up -- Registering flow nodes -- Registering your CryDev account -- What just happened? -- Running the sample application -- Editor -- Starting the Editor -- Launcher -- Starting the Launcher -- Dedicated server -- Compiling the CryGame project (C++) -- What just happened? -- The CE Game Programming Sample solution breakdown -- CryGame -- CryAction -- CryCommon -- The CryENGINE folder structure -- PAK files -- File query priority -- Attaching the debugger -- What just happened? -- Summary -- 2. Visual Scripting with Flowgraph -- Concept of flowgraphs -- Opening

the Flowgraph Editor -- A tour of the Flowgraph Editor -- Components -- Terminology -- Component categories -- Flowgraph types -- AI Actions -- UI Actions -- Material FX -- FG Modules -- Entities -- Prefabs -- Creating a flowgraph -- The flowgraph entity -- Spawning FlowgraphEntity -- Attaching a new flowgraph.
Adding nodes into flowgraphs -- Input and output ports -- Port types -- Target entities -- Linking flownodes -- Testing our flowgraph -- The stock flownode overview -- Building a clock -- Listening for player input -- Executing on a loop -- Flowgraph modules -- Creating a module -- Calling a module -- Module parameters/ports -- Custom flownodes -- Creating a custom node in C++ -- Organizing nodes -- Creating a new node file -- Breaking down of code -- The node functions overview -- Implementing GetConfiguration -- Creating ports -- Assigning arrays to the node configuration -- Flownode configuration flags -- Implementing ProcessEvent -- Creating a custom node in C# -- Adding inputs -- Adding outputs -- Implementing Activate -- Target entities -- Summary -- 3. Creating and Utilizing Custom Entities -- Introducing the entity system -- Entity classes -- Entities -- entityId -- EntityGUID -- Game objects -- The entity pool system -- Creating a custom entity -- Creating an entity using Lua -- Common Lua entity callbacks -- Creating an entity in C# -- Adding Editor properties -- Property folders -- Creating an entity in C++ -- Creating a custom entity class -- Implementing a property handler -- Entity flownodes -- Creating an entity flownode in Lua -- Creating an entity flownode using C# -- Creating an entity flownode in C++ -- Registering the entity node -- The final code -- Game objects -- Game object extensions -- Creating a game object extension in C++ -- Activating our extension -- Summary -- 4. Game Rules -- Introduction to game rules -- IGameRules interface - game rules -- Scripting - game modes -- Loading a level -- Implementing the game rules interface -- Registering the game object extension -- Creating custom game modes -- Scripting -- Lua scripting -- Invoking methods -- Invoking methods with parameters -- Getting values returned from Lua.
Getting table values -- CryMono scripting -- Calling methods -- Return values -- Properties -- Fields -- Creating a basic game mode in C# -- Defining our intention -- Creating the actor -- Spawning the actor -- Handling disconnections -- Assigning the player to a team -- Implementing Headquarters -- Adding the end game event -- Creating the Headquarters entity -- Detour - trigger bounds and entity areas -- Populating the level -- Summary -- 5. Creating Custom Actors -- Introducing the actor system -- Channel identifiers -- Actor spawning -- Removing actors -- The view system -- Linking views to game objects -- Creating custom actors -- Creating actors in C# -- The CryMono class hierarchy -- Using native and CryMono actors alongside each other -- C++ actor registration -- C# declaration -- Creating actors in C++ -- Registering actors -- Camera handling -- Implementing IGameObjectView -- Creating a top-down camera -- View rotation -- Player inputs -- The hardware mouse -- Action maps -- Listening for action map events -- IActionListener -- Enabling action map sections -- Animated characters -- Movement requests -- Adding a movement request -- The Mannequin animation system -- The Mannequin Editor -- Preview setup -- Creating contexts -- Creating fragments -- Adding options -- Creating and using tags -- Appending tags to Options -- Saving -- Starting fragments -- Acquiring the fragment identifier -- Queuing the fragment -- Setting tags -- Forcing actions into requerying options -- Debugging Mannequin -- Setting up Mannequin for a custom entity -- Initializing

Mannequin -- Loading the controller definition -- Setting the active context -- Summary -- 6. Artificial Intelligence -- The Artificial Intelligence (AI) system -- Scripting -- AI actors -- Goal pipes -- Creating custom pipes -- Selecting pipes -- Signals -- AI behaviors -- Sample -- IAIObject.
Creating custom AI -- Registering an AI actor implementation -- In C# -- In C++ -- Creating the AI entity definition -- AI behaviors and characters -- Understanding and using behavior selection trees -- Variables -- Signal variables -- Leaves / behavior queries -- Character -- NavigationType -- Creating custom behaviors -- Listening to signals -- AI base definition breakdown -- The AISample_x table -- The Properties table -- The AIMovementAbility table -- The CreateAI function -- The RegisterAI function -- Summary -- 7. The User Interface -- Flash movie clips and UI graphs -- Elements -- XML Breakdown -- Actions -- Creating a main menu -- Creating menu elements -- Exposing ActionScript assets -- Functions -- Events -- Variables -- Arrays -- Exposing MovieClip instances to flowgraph -- Creating the UI actions -- Creating the state control graph -- Creating the MainMenu action -- Adding buttons -- End result -- Engine ActionScript callbacks -- Creating UI game event systems -- Implementing IUIGameEventSystem -- Receiving events -- Dispatching events -- Dispatching the event -- Summary -- 8. Multiplayer and Networking -- The networking system -- Network identifiers -- Net channels -- Net nubs -- Setting up a multiplayer game -- Starting the server -- Dedicated server -- Launcher -- Connecting to a server via the console -- Debugging networked games -- Networking using game object extensions -- Remote Method Invocation (RMI) -- RMI structure -- Parameters -- Attach type -- Server/client separation -- Function definition -- RMI example -- Network aspect serialization -- Aspects -- Compression policies -- Creating a new compression policy -- Exposing Lua entities to the network -- Net.Expose -- Function implementation -- Invoking RMIs -- On the server -- On all clients -- On all other clients -- Binding our entity to the network -- Summary. 9. Physics Programming -- CryPhysics -- Physicalized entity types -- Introducing physical entity identifiers -- Drawing entity proxies -- Entity types -- Helper types -- Physical entity actions, parameters, and status -- Parameters -- Actions -- Status -- Physicalized entity type details -- Common parameters -- Common actions -- Common statuses -- Static -- Rigid -- Wheeled vehicle -- Unique parameters -- Unique statuses -- Unique actions -- Living -- Unique parameters -- Unique statuses -- Unique actions -- Particle -- Unique parameters -- Articulated -- Unique parameters -- Rope -- Unique parameters -- Soft -- Unique parameters -- Unique actions -- Ray world intersections -- The ray_hit struct -- Commonly used member variables -- Origin and direction -- Object types and ray flags -- Object types -- Ray flags -- Allowing multiple ray hits -- Creating a physicalized entity -- In C++ -- In C# -- Simulating explosions -- Summary -- 10. Rendering Programming -- The renderer details -- Shaders -- Shader permutations -- Shader cache -- PAK files -- Render nodes -- Rendering breakdown -- Pre update -- Post update -- Rendering new viewports using render contexts -- Rendering -- Using the I3DEngine::RenderWorld function -- I3DEngine::RenderWorld flags -- Shaders -- The shader description -- Texture slots -- Shader flags -- Material flags -- Engine flags -- Runtime flags -- Samplers -- Using texture slots with samplers -- Obtaining a texture -- Manipulating static objects at runtime -- Modifying materials at runtime -- Cloning a material -- Material parameters -- Shader parameters -- Example - Dynamic alpha-test for vegetation -- Summary -- 11. Effects and

	Sound -- Introducing effects -- Material effects -- Surface types -- Adding or modifying surface types -- Particle effects -- Sound effects -- FMOD Designer -- Creating and triggering material effects. Automated playback based on physical interactions.
Sommario/riassunto	This book provides you with step-by-step exercises covering the various systems of CryENGINE and comprehensively explains their workings in a way that can be easily understood by readers of any skill level to help you develop your very own CryENGINE games.This book is intended for developers looking to harness the power of CryENGINE, providing a good grounding in how to use the engine to its full potential. The book assumes basic knowledge of the engine and its editor in non-programming areas.