DOWNLOAD PDF CORE LANGUAGE ENGINE

Chapter 1 : PHP - Wikipedia

The Core Language Engine presents the theoretical and engineering advances embodied in one of the most comprehensive natural language processing systems designed to date.

Razor is a markup syntax for embedding server-based code into webpages. Files containing Razor generally have a. When an symbol is followed by a Razor reserved keyword , it transitions into Razor-specific markup. Otherwise, it transitions into plain C. To escape an symbol in Razor markup, use a second symbol: The email addresses in the following example are untouched by Razor parsing: If the C statement has a clear ending, spaces can be intermingled: The following code is not valid: All elements must be either self-closing or have a matching end tag. Did you intend to invoke the method? Explicit Razor expressions Explicit Razor expressions consist of an symbol with balanced parenthesis. Explicit expressions can be used to render output from generic methods in. The following markup shows how to correct the error shown earlier caused by the brackets of a C generic. The code is written as an explicit expression: Raw on unsanitized user input is a security risk. User input might contain malicious JavaScript or other exploits. Sanitizing user input is difficult. Raw with user input. Code blocks and expressions in a view share the same scope and are defined in order: Explicit Line Transition with: To render the rest of an entire line as HTML inside a code block, use the: Extra characters in a Razor file can cause compiler errors at statements later in the block. These compiler errors can be difficult to understand because the actual error occurs before the reported error. Control structures Control structures are an extension of code blocks. All aspects of code blocks transitioning to markup, inline C also apply to the following structures: Conditionals if, else if, else, and switch if controls when code runs: To render a list of people: Length ; Compound using In C , a using statement is used to ensure an object is disposed.

Chapter 2 : [PDF] The Core Language Engine Read Online - Video Dailymotion

The Core Language Engine (CLE) is a domain independent system for translating natural language (English) sentences into formal representations of their literal meanings which are capable of.

The highest-performing shader-based rendering available to game developers helps you quickly produce even the most complex scenes quickly and efficiently. Author shaders for the Direct3D pipeline using up to shader model 3. Existing techniques include all manner of sophisticated lighting effects from the non-photorealistic rendering NPR style of Team Fortress 2 to the hyper-realistic look of the Half-Life 2 episodesâ€”all in one engine. LOD on Models and World. Achieve maximum performance on all levels of hardware with automatic management of geometric Level of Detail LOD. Enables antialiasing of alpha-tested primitives such as foliage, fences, and grillwork. Use distance-coded alpha masking for infinite resolution texture maskingâ€”useful for resolution-independent UI elements or any alpha-tested primitives. Dynamic Lighting and Shadows Radiosity Lighting. World geometry is lit with radiosity lighting stored in light maps or per vertex to provide immersive environments. Light maps encode directional information so that lighting combines naturally with bump maps, resulting in more accurate lighting of local surface detail, including self-shadowing. Radiosity is computed using a distributed radiosity solver vrad which can be deployed across your local network for rapid iteration on world lighting. All lighting data including light maps, environment maps and dynamic lights in the scene are computed in high dynamic range space for natural lighting. High dynamic range lighting is supported in the Source engine on all DirectX 9 level hardware, unlike most competing engines. Dynamic objects and characters in the world pick up bounced light computed during offline radiosity computation. This lighting effect truly grounds characters and other dynamic objects in the game world. High Performance Dynamic Shadows. Dynamic objects and characters in the world generate high performance shadows which are projected onto world geometry, providing critical lighting cues. Shadow depth textures allow for realistic self-shadowing of objects in the world, providing a more realistic and immersive effect, at a greater cost than RTT shadows. A separate rim lighting term can be used to highlight key characters, as in Left 4 Dead, or provide a stylized look, as in Team Fortress 2. Apply diffuse, specular, detail, emissive, iridescent and other special effects. Effects Full Range of Special Effects. Including particles, beams, volumetric smoke, sparks, blood and environmental effects like fog and rain. Multicore graphics optimizations improves particle rendering performance. Edit and create particle systems with fully interactive preview and the ability to see edited systems immediately in the game. Particle shaders use scene depth information to eliminate hard intersections seen in traditional particle rendering. Render full-screen camera motion blur in real-time. Generate realistic-looking reflective water surfaces with refraction and Fresnel effects. Source defines sets of materials that specify what the object is made from and the texture used for that object. This system is much more flexible than other texture-only based systems. Self-shadowed Bump Maps create soft shadows and ambient occlusion with both dynamic and pre-calculated radiosity lighting. Source renders self-shadowed bump maps on both current and older-generation graphics hardware. Particularly useful for facial and clothing wrinkles, additional texture maps are blended in to provide dynamic surface detail in areas of models which compress and stretch. Combine low frequency textures with high frequency detail to conserve video memory while maintaining apparent texture density. Define blend masks with variable sharp edges, combine textures using multiple different modes, and apply per-surface color correction. Interactively edit the color cast and contrast of your scene to match the desired art style. Modeling and Animation Compatibility with popular graphics and 3D modeling software lets you model realistic or highly-stylized characters, weapons, vehicles, and props within the Source environment. A robust set of tools help you efficiently rig, animate, and define physics interactions for a wide range of characters and objects. Advanced Character Meshes Create believable characters with accurate human characteristics: Simulated musculature projects character emotions, speech, and body language. Characters accurately and naturally lip-synch speech in any language. Improved human skin shading. Skin rendering includes natural-looking Phong shading, including a view-dependent Fresnel effect tuned for realistic human skin. Streamline artist workflow by

integrating Source modeling functions with familiar programs favored by creative professionals: Compile models and materials with the Studiomdl and Vtex tools. Preview models in wireframe, shaded, or textured view modes; set up hit boxes, play animations, and fine-tune physics constraints. Advanced Procedural Animation Tools. Tune ragdoll, animated bone followers, and custom procedural physics controllers. Seamlessly blend gestures to create continuous movement or add depth to any character situation. Synthesize animations out of multiple pieces. Easily craft speech and emotions using the Faceposer facial expression tool. Environments Valve Hammer Editor, the Source map creation tool, is an intuitive design environment for constructing level architecture geometry, texturing, lighting ; placing and scripting models, entities, and NPCs; and compiling and running game levels. Freely create and sculpt natural hills, valleys, trenches, slopes, and tunnels using displacement geometry brush surfaces. Transform and clip displacement surfaces using brush and vertex tools. Define how objects interact with world architecture using a comprehensive inventory of brushes, including occluders, doors, triggers, area portals, soundscapes, and more. Cube and environment mapping skyboxing extends the horizon and adds parallax on distant objects. Easily view the effects of manipulating light sources within a level. Integrated within the Valve Hammer Editor, the Model Browser helps you quickly find, preview, and place models. Construct responsive, transformable worlds where AI characters interact with physically simulated objects, and sound and graphics follow from physics. Construct detailed machinery with functioning mechanisms, gears, belts, and pulleys. Non-player characters and, in a multiplayer game, other players can come along for a ride in cars that feature realistic suspensions with springs on each wheel and wheels that slip and skid depending on the surface material. Leaning during acceleration, deceleration, and turning enhances the realism of the driving experience. Physics-based animation simulates flexible hair and clothing and blends ragdoll physics with predefined animations. Ropes and Cables can be dynamically shaken or broken by level events. Realistically define bone movement within a physical system using a series of constraints that enable a nearly limitless range of complex movements. Game Mechanics Innovative and seamless interaction between player and non-player characters is a hallmark of Source-powered games. A sophisticated AI system allows NPCs to follow players, join in the fight, or engage the player in epic non-scripted battles. AI characters can run, fly, jump, crouch, climb stairs, and burrow underground. The sophisticated Pathfinding and Navigation System features a local avoidance system to help NPCs navigate around obstacles; a path cost system for fine tuning pathfinding choices; and dynamic and level designer-controlled path connections. The system recognizes the size of creaturesâ€”knowing where they can and cannot travelâ€”and automatically accounts for all known methods of movement. The AI Sensory System emulates human sensesâ€—vision, hearing, and even scentâ€—to track and identify objects. The tunable system can be used for nearly limitless in-game mechanics and player challenges. Set a relationship between charactersâ€—an NPC, an NPC class, or player s â€—to influence behavior and action based on a variety of entities, such as hate, like, or fear. Squads of AI characters can operate together and share knowledge about enemies. AI squad members track the status of each otherâ€—while one squad member is laying cover fire, another may recognize the opportunity to move or reload a nearly-empty gun cartridge. Licensees of Source technology have access to all of the source code that Valve uses to build the Source engine as well as our award-winning games. This allows your development staff to spend its time realizing your game design rather than starting from the scratch. Source engine games utilize multi-core processors in both the PC and XBox to deliver high-performance gaming experiences. Make the modifications you need for your game to any part of the codebaseâ€—Source licensees have access to it all. Take advantage of the memory and performance optimizations that Valve has developed for its own products. Audio Source includes a powerful suite of digital audio technologies to create vibrant, dynamic soundscapes within the game world. Seamlessly blend sound effects, dialogue, and music with visual elements for an intensely immersive and realistic playing experience. Sound mixing processing can run on separate cores on multicore systems. Sounds can be asynchronously loaded, streamed, cached into a fixed pool or preloaded. Support for stereo, headphone, 4 channel and 5. Apply occlusion and obstruction effects, distance, and environmental reverberation using custom 3D spatialization algorithms. Custom or preset DSP effects can be triggered in-game based on environmental geometry. Sound designers can author custom doppler shift, distance cueing and other multichannel effects.

DOWNLOAD PDF CORE LANGUAGE ENGINE

Real-time Wave File Stitching. A simple scripting system lets audio designers build aggregate effects out of wave sequences. Audio designers can build several overall mixes using scripts and the game can dynamically switch between those mixes to get different fader settings for action vs. Scripting System for Environmental Sounds. This can also be modified by game logic for dynamic effects. Audio designers can define scripts to set up material specific impact, friction, and footstep effects as well as reflection parameters for automatic room DSP. Prediction Analysis and Server-Side Lag Compensation for reducing the visible effects of latency in the connection between clients and the game server. Displays all active game servers and allows a player to choose which one to participate on. Players can filter and sort server lists in order to speed up the display and selection of a server. Allows players to message each other both in and out of the game, as well as join friends in existing games. No more confusion about what server friends are on or how to chatâ€”the Friends Instant Messenger keeps everyone connected. Custom tools to convert PC assets maps, models, materials to compliant formats.

Chapter 3 : SteelSeries Engine Software - GameSense & Customization | SteelSeries

The Swedish Core Language Engine (S-CLE) is a Swedish version of the Core Language Engine (Alshawi [1]) and is described in GambÃ¤ck and Rayner [7].

Implementing a Text Templating Language and Engine for. NET November 13, C ,. Net comments edit Last year, before developing markdig , I spent a few weeks developing a brand new text templating language and engine called scriban. The post is organized like this: The documentation and tests part gives more detail about how the documentation was handled and how it contributes to the tests as well. The benchmarks by comparing scriban with various existing solutions Why? NET, I have been using text templating in the - non exhaustive - list of the following endeavors: When I started this project, I used at that time the Microsoft T4 templating and I wrote a couple of templates with it. In order to simplify the deployment of the code generator, I had to use the Mono. The compilation process of templates into assemblies is usually way too slow and the syntax is a bit cumbersome to work with C based. SharpDoc was also my attempt a few years ago to bring documentation generation for. It was actually running a lot more faster than sandcastle, bringing more opportunities for overriding templatesâ€¦etc. Though, it is still used by MonoGame for their documentation. For this project, I developed at that time SharpRazor which is the much more lightweight equivalent of RazorEngine. It was actually pretty nice to be able to step into the templates with debugging RazorEngine was not exposing this at that time, and it was a pain to bring this feature to it, mainly because the code was layered with tons of interfaces that were actually abstracting way too much the core Razor Engine. Again, it is sharing many similarities with T4 C syntax for example and specially the fact that the time to generate the assemblies for the templating was slowing down the iteration process. For this website itself, when I migrated my blog from blogger around two years ago to use instead Jekyll and GitHub pages. It has been simplifying a lot the pain of updating this blog and giving me a lot more freedom to control the layout and templating of this website. Static website generators have been here for years if not decades and departing from CMS solutions was a real liberation for me which I unfortunately used for too many years with solutions like Joomla PHP as much as it improved the editing workflow with git versioningâ€¦etc. NET components I was looking for. Hence, I decided that I would develop these core parts first -. But that was of course way more work than I expected! Requirements As for markdig, I settled on a few key features I wanted to see in a text templating language and engine, both in terms of language and runtime requirements: Liquid templates using DotLiquid: With my experience with liquid templates in Jekyll, while I liked the simplicity of the language, I felt also too many times frustrated by its limitations. It looked not built from the ground-up as a well-thought language, but more like a hack language, not really specified in many parts. I discovered also that it was actually so under specified that custom tag implementations would allow different syntaxes in the language that would be incompatible with other liquid implems. I have described this in the Known issues of liquid support in scriban you read this well, there is a compatibility mode to parse Liquid templates in scriban! More details further down! Mustache templates using Nustache: NET, I was also looking for a more simple and efficient integration. I also looked a bit outside the. NET world at what other platforms were providing: MyTitle Jinja in Python this one was probably the one I liked the most, and by looking at it again while writing this blog post, I have likely been influenced by it too when choosing the syntax for scriban. I got a even bit influenced also by Javascriptâ€¦ nooooo!!! NET runtime I was looking for a: NET assembly but could be brought later while still being competitive Safe runtime, typically to be able to host end-user scripts in a shared environment Providing a proper control on the AST of the code, eventually with the ability of saving back the AST to original source code in order to be able to translate, or manipulate scripts programmatically Easy to integrate with. NET were enough efficient. But looking quickly at the few options at that time, most of the implementations were often using regex based parsers which are very slow and their codebase were not carefully crafted to avoid GC allocations at parsing and rendering time. The language syntax For the syntax, I wanted something very familiar e. For this project, I took very seriously the need for a documentation, so you will find a lot more details in the scriban language documentation. In the example above, this mode would

strip the entire line trim space on the left and trim space on the right up to the first newline. I have bring also a few practical stuffs in the language that I have been looking for a lot in liquid: You will find more details for a comparison of liquid with scriban in the liquid-support documentation Custom functions One limitation I hit quite often with liquid was its inability to declare local functions directly into the language to compact very simple expressions that you want to express with a single call. Suppose that you would want to make a function that would downcase a text, truncate to 15 characters and upper case the first letter and use it in multiple place in your template, in scriban you could declare a function to do this: This allow to compose your functions in a very powerful ways! There are also anonymous function expressions with the do Of course, recursive functions are possible, and the runtime protects scripts from running too deeply with an exception reporting where a recursive call is exceeding the defined limit. I wrote also a Scriban syntax colorizer for Visual Studio Code so that you can enjoy a bit of syntax highlighting with scriban templates. Engine details The power of scriban is not only defined by its flexible language but also by the efficiency of its runtime for both: The lexer is the class responsible for transforming a flow of characters into a flow of language tokens e. The parser is the class responsible for transforming this flow of language tokens into a full Abstract Syntax Tree AST of the original script template, a runtime representation of it. The lexer In order to push the performance of the templating system, GC allocations have been carefully tracked. Typically for the lexer, we have a zero GC allocation lexer that is only producing a series of token which is a struct through a custom IEnumerable iterator also a struct so that we are able to get a very minimal memory footprint. A token is simply a struct with the following information: Type of the token identifier, numberâ€¦etc Precise starting character offset from the source document, including also precise line and column. Precise ending character offset from the source document, including also precise line and column. Though there is a particularity of the scriban lexer and parser: There is a mode to parse actual liquid templates directly into a scriban AST and runtime But also, cherry on the top of this, there is a mode in the parser to precisely keep all symbols parsed, and to save the AST back to code, meaning that scriban not only can execute liquid templates, but can also translate and save back liquid templates to scriban templates, allowing to transition your existing templates to scriban very easily. This mode of saving back a scriban AST to text can open lots of opportunities, like the ability to modify a scriban template programmatically and save it back to the disk. This round trip mode has been tested quite extensively in scriban, so that all scriban test templates and liquid tests are actually parsed, run, re-saved to code, re-parsed and re-run to check that: Obviously, this one, we want a strict correspondence and not an almost! To my knowledge, at least in the. NET ecosystem, scriban is the only text templating language and engine that allows such a scenario of modifying the AST in memory and saving it back to the disk, in the addition to support liquid templates as-is and translate them to a new more powerful runtime. Rendering Once the AST is in memory, we can interpret each syntax node. This is one of the easiest part of the engine, as all the information has been largely processed by the parser, so we just have to execute instructions as they are listed in each syntax node. Each node in an AST implements the following method: Though, compared to Razor, a scriban template similar a liquid template can bring a bit more practical helpers to help you write templates. For example, typically a for loop provides many builtin variables accessible from the loop:

Chapter 4 : Implementing a Text Templating Language and Engine for .NET | xoofx

The Core Language Engine begins by describing the data structure that is output by the system, Logical Form (Chapter 2), followed by descriptions of syntax and semantics rules (Chapters 3, 4, and 5).

The following "Hello, World! This short delimiter makes script files less portable, since support for them can be disabled in the local PHP configuration and it is therefore discouraged. Variables are prefixed with a dollar symbol , and a type does not need to be specified in advance. However, before PHP 7. In terms of keywords and language syntax, PHP is similar to the C style syntax. Data types[edit] PHP stores integers in a platform-dependent range, either a bit or bit signed integer equivalent to the C-language long type. Unsigned integers are converted to signed values in certain situations; this behavior is different from other programming languages. Floating point numbers are also stored in a platform-specific range. They can be specified using floating point notation, or two forms of scientific notation. These are typically created by functions from a particular extension, and can only be processed by functions from the same extension; examples include file, image, and database resources. Order is preserved in lists of values and in hashes with both keys and values, and the two can be intermingled. Custom functions may be defined by the developer, e. In this manner, normal PHP functions can be used, for example, as callbacks or within function tables. Function calls must use parentheses, with the exception of zero-argument class constructor functions called with the PHP operator new, in which case parentheses are optional. Such a function is a first-class object, meaning that it can be stored in a variable, passed as a parameter to other functions, etc. Object handling was completely rewritten for PHP 5, expanding the feature set and enhancing performance. In the new approach, objects are referenced by handle , and not by value. PHP 5 introduced private and protected member variables and methods, along with abstract classes , final classes , abstract methods , and final methods. Furthermore, PHP 5 added interfaces and allowed for multiple interfaces to be implemented. There are special interfaces that allow objects to interact with the runtime system. Objects implementing ArrayAccess can be used with array syntax and objects implementing Iterator or IteratorAggregate can be used with the foreach language construct. There is no virtual table feature in the engine, so static variables are bound with a name instead of a reference at compile time. For convenience, the engine will supply a function that imports the properties of the source object, so the programmer can start with a by-value replica of the source object and only override properties that need to be changed. The default is public, if only var is used; var is a synonym for public. Items declared public can be accessed everywhere. To disambiguate it from other implementations, it is sometimes unofficially called "Zend PHP". The Zend Engine compiles PHP source code on-the-fly into an internal format that it can execute, thus it works as an interpreter. Due to the complex and nuanced semantics of PHP, defined by how Zend works, it is difficult for competing implementations to offer complete compatibility. Alternative implementations include the following: Numerous functions familiar to C programmers, such as those in the stdio family, are available in standard PHP builds. Numerous extensions have been written to add support for the Windows API , process management on Unix-like operating systems , multibyte strings Unicode , cURL , and several popular compression formats.

Chapter 5 : Razor syntax reference for calendrierdelascience.com Core | Microsoft Docs

LEXICAL ACQUISITION IN THE CORE LANGUAGE ENGINE David M. Carter SRI International Cambridge Research Centre 23 Millers Yard, Mill Lane.

Carmack commented on his blog in that "there are still bits of early Quake code in Half-Life 2". Over the next few years, we used these terms internally as "Goldsource" and "Source". At least initially, the Goldsrc branch of code referred to the codebase that was currently released , and Src referred to the next set of more risky technology that we were working on. When it came down to show Half-Life 2 for the first time at E3 , it was part of our internal communication to refer to the "Source" engine vs. Among others, Source uses Bink Video for video playback. Different systems within Source are represented by separate modules which can be updated independently. With Steam , Valve can distribute these updates automatically among its many users. In practice, however, there have been occasional breaks in this chain of compatibility. The release of Half-Life 2: Episode One and The Orange Box both introduced new versions of the engine that could not be used to run older games or mods without the developers performing upgrades to code and, in some cases, content. This was demonstrated in , when Valve updated all of their core Source games to the latest engine build. A screenshot of Half-Life 2: The high dynamic range rendering and Phong shading effects are evident. HDR rendering and color correction were first implemented in using Day of Defeat: Episode One introduced Phong shading and other smaller features. Since the transition to Steam Pipe, this branch was made deprecated and is now used for backward compatibility with older mods. It was mentioned again by Gabe Newell in as a piece of technology he would like to add to Source to implement support for much larger scenes that are impossible with strictly polygonal objects. An artist-driven, threaded particle system replaced previously hard-coded effects for all of the games within. In addition, the facial animation system was made hardware-accelerated on modern video cards for "feature film and broadcast television" quality. It includes asset converters, cross-platform play and Xbox Live integration. Gabe Newell cited these issues when criticizing the console during the release of The Orange Box. Multiprocessor support was further expanded, allowing for features like split screen multiplayer, additional post-processing effects, event scripting with Squirrel , and the highly-dynamic AI Director. The menu interface was re-implemented with a new layout designed to be more console-oriented. This branch later fueled the releases of Alien Swarm and Portal 2 , the former released with source code outlining many of the changes made since the branch began. Portal 2, in addition, served as the result of Valve taking the problem of porting to PlayStation 3 in-house, and in combination with Steamworks integration creating what they called "the best console version of the game". Valve announced that all their future games will be released simultaneously for Windows and Mac. The game code to these branches was made public to mod developers in , and they serve as the current stable release of Source designated for mods. It comes with several command-line programs designed for special functions within the asset pipeline, as well as a few GUI-based programs designed for handling more complex functions. Source SDK was launched as a free standalone toolset through Steam, and required a Source game to be purchased on the same account. The three applications mentioned below are now included in the install of each game. There are three applications packaged in the Source SDK: The tool was originally known as Worldcraft and was developed independently by Ben Morris before Valve acquired it. Developers may use the program to view models and their corresponding animations, attachment points, bones, and so on. Face Poser is the tool used to access facial animations and choreography systems. This tool allows one to edit facial expressions , gestures and movements for characters, lip sync speech, and sequence expressions and other acting cues and preview what the scene will look like in the game engine. It can be launched through Windows or Linux, and can allow for custom levels and assets. Most third-party servers additionally run Metamod: Source, but is more associated with Team Fortress 2. Today, it is open for public use and downloadable via the Steam client. These new articles covered the previously undocumented Counter-Strike:

Chapter 6 : The Swedish Core Language Engine - SODA

DOWNLOAD PDF CORE LANGUAGE ENGINE

Engine Engine Number 9 - 3D Animation - English Nursery rhymes - 3d Rhymes - Kids Rhymes - Rhymes for childrens.

Chapter 7 : Source Engine Features - Valve Developer Community

The paper describes a Swedish-language customization (S-CLE) of the SRI Core Language Engine, A shorter version of this paper appears in calendrierdelascience.comerg (ed.): Papers from the Third Nordic Conference on Text Comprehension in Man and Machine, Link Sweden, which has been developed at SICS from the original English-language version by replacing English-specific modules with corresponding.

Chapter 8 : CiteSeerX â€" The Swedish Core Language Engine

The paper describes a Swedish-language customization (S-CLE) of the SRI Core Language Engine, which has been developed at SICS from the original English-language version by replacing English-specific modules with corresponding Swedish-language versions. The S-CLE is intended to be used as a building.

Chapter 9 : The Core Language Engine - CORE

Abstract. We present ECSGlasses: eye contact sensing glasses that report when people look at their wearer. When eye contact is detected, the glasses stream this information to appliances to inform these about the wearer's engagement.