CryENGINE Startup

This article walks through the process of starting a CryENGINE application.

Note: The CryENGINE Free SDK utilizes CryDevLogin protection. CrySystem will not function without obtaining the necessary login details.

Initialization

1 Prior to engine initialization, the Launcher application has to set the working directory to the root of the CryENGINE installation. By default, the working directory is the directory in which the executable is placed. See Win API function SetCurrentDirectory.

2 The application should then load the game dll, and invoke **CreateGameStartup** in order to initialize the engine and obtain necessary subsystem instances.

The launcher application passes SSystemInitParams to the engine, which is distributed amongst the different modules. This allows for easily passing commonly used startup data, as well as configuration variables to for example disable the renderer.

Example

```
auto hCryGame = CryLoadLibrary(“CryGame.dll”); // Hardcoded, if one wants to utilize
sys_dll_game a cfg parser has to be written to extract the new value from system.cfg.

// if library load failed, hCryGame will be null, make sure to check for this.
if(!hCryGame)
	return;

auto entryFunc = (IGameStartup::TEntryFunction)CryGetProcAddress(hCryGame,
“CreateGameStartup”);

// if entryFunc is null, the game dll is incompatible.
if(!entryFunc)
{
	CryFreeLibrary(entryFunc);
	return;
}

SSystemInitParams startupParams;
```

```
startupParams.hInstance = GetModuleHandle(0);
startupParams.sLogFileName = “MyCryENGINEApp.log”;
// strcpy(startupParams.szSystemCmdLine, cmdLine); pass application command line
arguments to the engine

IGameStartup *pGameStartup = entryFunc(); // call CryGame.dll’s CreateGameStartup
method.
if(!pGameStartup)
{
	CryFreeLibrary(entryFunc);
	return;
}

if(pGameStartup->Init(startupParams))
{
	// IGameStartup::Run handles the engine loop, when it returns it’s time to
close the application.
	pGameStartup->Run(NULL); // Don’t automatically load a level.
}

// When IGameStartup::Init is finished
pGameStartup->Shutdown();
pGameStartup = 0;

CryFreeLibrary(entryFunc);
```

Update loop

The update loop is done via IGameStartup::Run, in the case of the default CryGame dll see CGameStartup::Update, which in turn calls CGame::Update.

CGame::Update handles updating of the game framework (which in turn handles per-frame updates of all other non-CryGame systems) each frame, via IGameFramework::PreUpdate and IGameFramework::PostUpdate.