Cog2D Documentation

Engine Version: Alpha 1  
Author: Erik Paldanius  
Contact: [erik.paldanius@gmail.com](mailto:erik.paldanius@gmail.com)

Content Assemblies

### Definition

A Content Assembly is a .NET assembly that references Cog2D’s main assembly (Cog2D.dll) and therefore may or may not contain types relevant to the game.

### Purpose

Cog2D will iterate through the types of loaded content assemblies when it’s initialized in order to prepare for their use during gameplay by doing expensive operations before the game has started, this in order to avoid slow-downs during gameplay.

### Limitations

All Content Assemblies must be loaded into the current AppDomain before Cog2D is initialized, otherwise their types will not be registered as valid components of the game which may result in crashes.

### Misc

-

Game Objects

### Definition

A Game Object is a .NET type that references ”Cog.Modules.Content.GameObject”.

### Purpose

Game Objects are meant to provide the functionality of a physical object that exists inside of a 2D environment (Scene).

### Limitations

All Game Objects must be attached to a

### Misc

Networking

### Network Messages

Network Messages are types that derive from Cog.Modules.Networking.NetworkMessage.

Types that derive from NetworkMessage may not be generic.

Types that derive from NetworkMessage may only contain types that implement *Cog.ISerializable*, is registered through *Cog.TypeSerializer* or is an Enum.

For performance reasons we recommend separately registering types through Cog.TypeSerializer instead of implementing Cog.ISerializable.

The following types are implemented by default:

* System.Boolean (bool)
* System.Byte (byte)
* System.SByte (sbyte)
* System.Int16 (short)
* System.UInt16 (ushort)
* System.Int32 (int)
* System.UInt32 (uint)
* System.Int64 (long)
* System.UInt64 (ulong)
* System.Single (float)
* System.Double (double)
* System.Decimal (decimal)
* System.Char (char)
* System.String (string)
* Cog.Vector2 (Vector2)
* Cog.Rectangle
* Cog.Color (Color)

Cog.TypeSerializer automatically creates array-variants of registered types using an unsigned short length indicator, registering your own custom array serializer is not supported.

### Strings

In many games string data takes up the majority of the data being sent, to counter this Cog2D provides a way to describe strings, **namely that attribute thingie.**

It is also to be noted that null strings maps to empty ones when serialized.

Todo

* Send / synchronize client permissions
* Add empty initialize overload
* Implement NetworkMessage enum array support
* ISerializable array support (?)
* TypeSerializer.GameObject (Reference?)
* Write networking documentation
* Research word C# code
* Write getting started

var type = new Program[5].GetType();

Console.WriteLine(type.IsArray);

Console.WriteLine(type.GetElementType().FullName);

Console.ReadKey();