Glossary of Terms

BigWorld Technology 1.9.1. Released 2008.

Software designed and built in Australia by BigWorld.

Level 3, 431 Glebe Point Road
Glebe NSW 2037, Australia
www.bigworldtech.com

Table of Contents

Chapter 1. Glossary

The list below describes some of the terms used throughout the BigWorld documents:

- **AOI**

  or, Area Of Interest: the physical region in which an entity is interested.

  Typically, this is a vertically oriented cylinder with a radius of 500 metres around the entity.

  The length of the radius is also known as the ghost distance, because it matches the distance between an entity and a nearby cell in which the entity needs to be ghosted. For more details, see the document Server Overview's section Server Components → CellApp → Real and Ghost Entities.

  The distance should be slightly larger than the distance that an avatar can see.

- **Area of Interest**

  See AOI.

- **Avatar**

  An entity associated with a character. This may be a Player Character (PC) or a Non-Player Character (NPC).

- **Base**

  An object used to represent the part of an entity that does not move.

  Unlike the part of an entity on a cell, which can move between different machines, a base does not move, thus giving a fixed location to talk to the entity.

- **Bundle**

  A bundle is a collection of messages. It maps to one or more UDP packets.

- **Call**

  A type of event: a remote method invocation. Differs from other types of event in that a call is made directly on a known object.

- **Camera**

  A camera is a view into the world. Cameras have a position and a viewing direction, stored in a matrix.

  There are a number of cameras that move in different ways, but exactly one must be active for each frame.

  The active camera matrix is communicated to the 3D Engine (Moo) before drawing the frame.

- **Chunk**

  The terrain, scene, and collision scene are split up into chunks that are loaded when necessary.

  At every frame, the client determines if any unloaded chunks may soon be visible, and instructs the loading thread to load them. Chunks are also disposed when no longer needed so the memory they use can be reclaimed.

- **Chunking**

  Term used for the concept of breaking the world into connected pieces.

It is usually used when talking about rendering the scene or loading world data, e.g.: 'The chunking of the terrain allows it to load faster.'

- **Collision scene**

  The collision scene contains a representation of the space that is used in physics calculations, in particular for calculating collisions between moving objects and the scenery, and querying properties about the point of impact.

- **Console**

  A console is a layer of text drawn over the scene and the GUI. Consoles are mainly used for debugging, but can also be used by scripts.

- **Detail object**

  A detail object is a small mesh drawn in large numbers onto the terrain around the camera position. They are separated from ordinary models for efficiency. The selection and placement of these objects is largely automatic, based on the texture of the underlying terrain.

- **Entity**

  An entity is the basic addressable object in the BigWorld system. The messages, sent between the BigWorld server and client, are all in the context of some entity.

  Entities have a type and a position, and a number of other properties determined by the type. Each entity type has a Python script class associated with it, and each entity instance in the world has an instance of its Python class to control it.

  Entities have no intrinsic representation in the world; rather their controlling script explicitly adds objects to the scene under the entity's authority.

  The BigWorld client presents a rich scripting environment for the implementation of entities, by exposing powerful classes and objects as Python types and modules. For more details, see the document Server Overview's section Server Components → CellApp → Scripting and Entities.

- **Event**

  A remote method invocation (see Call) or a property update.

- **Ghost**

  A non-authoritative representation of a real object.

- **GUI element**

  A range of Graphical User Interface (GUI) elements may also exist in the world, without being added to the scene, but being overlaid after the scene is drawn.

  Scripts have access to these elements.

- **Light**

  A light is a scene object that illuminates models. Like models, lights must be in the scene to have an effect. Scripts also have access to lights.

- **Message**

  A message represents a single event or call.

- **Model**

  A model is a mesh, possibly attached to a skeleton that can be drawn in the world.

  Models often have animations (especially those with skeletons), which affect the configuration of the mesh. In order for a model to be properly updated and drawn, it must be put into the scene.

  Entity scripts make use of models (often just one) to represent themselves in the world.

- **Object**

  The most general concept of a thing in the game. For example, a button or an object that represents a team are examples of objects. It maps to the C++ concept of an object.

- **Particle system**

  A particle system displays a moving and changing set of particles. These do not exist in the scene by themselves, but are attached to a model instead.

  Scripts also have access to particle systems, and can construct a great variety of graphical effects by specifying the rules for a system.

- **Personality script**

  The personality script is a global script that determines the game-specific top-level behaviour of the client, base and cell applications. It remains in scope for the entire running of the application, so for example on the client, regardless of changes in the active player, the personality script remains the same.

  The main job for personality scripts is to handle global events, for example for initialisation, the handling of input events, or responding to when a server process has fully started up, or the geometry is fully mapped. Additional tasks that are appropriate for the personality script are tasks such as managing the active camera, and handling global user interface features such as a chat console.

- **Player**

  The player is the special name given to the entity controlled by the user. This entity is usually an avatar.

  This is the only entity that the BigWorld server will let the client control - attempts to control others will be denied by the server.

  Also, the client bandwidth (data coming from the server) allocated to each entity is proportional to that entity's proximity to the client's player entity. For details on bandwidth management and load balancing, see the document Server Overview.

- **Player script**

  Each entity type that might become a player also provides a player script in Python as a class that inherits from the ordinary Python entity class.

  The player script has the important job of handling the user input and graphical feedback specific to that player type. This involves everything from entity movement (on the input side) to the display of the current item (on the graphical feedback side). A number of C++ classes are provided as Python types to aid the player script in these duties.

- **Portal**

  Chunks are connected to each other through portals.

- **Real**

  The authoritative representation of an object (at least within the scope of the cells).

- **Scene**

  The scene is a concept encompassing the majority of objects that have a three-dimensional physical manifestation in the world.

  The static base of the scene can be constructed using tools such as WorldEditor. Other scene objects are added dynamically based on information from the server.

  The scene is sometimes referred to as the scene graph.

- **Terrain**

  The terrain is the primary geometric object that gives overall form to the surface of the world. Most other objects in the world are placed on top of the terrain.

  It is implemented as a large contiguous height-mapped mesh, partitioned into chunks for efficient processing.

  Textures are applied to the terrain and are merged using alpha blending.

- **World**

  The world refers to the entire scene, and is made up of many diverse components such as chunks, terrain, camera, and entities.

  Most of these components are updated and drawn at every frame. More importantly, the world defines a fixed geographic reference frame, as opposed to the camera or objects, which have their own movable reference frames.