Setting Up a MasterServer- Part 2- NetworkViews and state Serialization

**NetworkViews and state serialization** **In Unity, game objects that need to be networked have a NetworkView component. The NetworkView component handles communication over the network, and even helps make networked state serialization easier. It can automatically serialize the state of a Transform, Rigidbody, or Animation component, or in one of your own scripts you can write a custom serialization function.** **When attached to a game object, NetworkView will generate a NetworkViewID for NetworkView. This ID serves to uniquely identify a NetworkView across the net- work. An object can be saved as part of a scene with NetworkView attached (this can be used for game managers, chat boxes, and so on), or it can be saved in the project as a prefab and spawned later via Network.Instantiate (this is used to gen- erate player objects, bullets, and so on). Network.Instantiate is the multiplayer equivalent to GameObject.Instantiate — it sends a message over the network to other clients so that all clients spawn the object. It also assigns a network ID to the object, which is used to identify the object across multiple clients (the same object will have the same network ID on every client).**

**Note** **A prefab is a template for a game object (such as the player object). You can use the Instantiate methods to create a copy of the template in the scene.**  **Spawned network game objects can also be destroyed via Network.Destroy. It is the multiplayer counterpart of GameObject.Destroy. It sends a message to all clients so that they all destroy the object. It also deletes any RPC messages associated with that object.** **NetworkView has a single component that it will serialize. This can be a Transform, a Rigidbody, an Animation, or one of your own components that has an OnSerial- izeNetworkView function. Serialized values can either be sent with the Reli- ableDeltaCompressed option, where values are always sent reliably and com- pressed to include only changes since the last update, or they can be sent with the Unreliable option, where values are not sent reliably and always include the full**

**NetworkViews and state serialization**

**In Unity, game objects that need to be networked have a NetworkView component. The NetworkView component handles communication over the network, and even helps make networked state serialization easier. It can automatically serialize the state of a Transform, Rigidbody, or Animation component, or in one of your own scripts you can write a custom serialization function.**

**When attached to a game object, NetworkView will generate a NetworkViewID for NetworkView. This ID serves to uniquely identify a NetworkView across the network. An object can be saved as part of a scene with NetworkView attached (this can be used for game managers, chat boxes, and so on), or it can be saved in the project as a prefab and spawned later via Network.Instantiate (this is used to generate player objects, bullets, and so on). Network.Instantiate is the multiplayer equivalent to GameObject.Instantiate — it sends a message over the network to other clients so that all clients spawn the object. It also assigns a network ID to the object, which is used to identify the object across multiple clients (the same object will have the same network ID on every client).**

**Note**

**A prefab is a template for a game object (such as the player object). You can use the Instantiate methods to create a copy of the template in the scene.**

**Spawned network game objects can also be destroyed via Network.Destroy. It is the multiplayer counterpart of GameObject.Destroy. It sends a message to all clients so that they all destroy the object. It also deletes any RPC messages associated with that object.**

**NetworkView has a single component that it will serialize. This can be a Transform, a Rigidbody, an Animation, or one of your own components that has an OnSerializeNetworkView function. Serialized values can either be sent with the ReliableDeltaCompressed option, where values are always sent reliably and compressed to include only changes since the last update, or they can be sent with the Unreliable option, where values are not sent reliably and always include the full**

**Book Navigation**

**values (not the change since the last update, since that would be impossible to pre- dict over UDP). Each method has its own advantages and disadvantages. If data is constantly changing, such as player position in a first person shooter, in general Unreliable is preferred to reduce latency. If data does not often change, use the Reli- ableDeltaCompressed option to reduce bandwidth (as only changes will be seri- alized).** **NetworkView can also call methods across the network via Remote Procedure Calls (RPC). RPCs are always completely reliable in Unity Networking, although some networking libraries allow you to send unreliable RPCs, such as uLink or TNet.**