TNet 3: Instantiating Objects With Parameters

When working on a game you often want to create a new object but need to specify some initial parameters. This can be done by extending the `TNManager.Instantiate` function with a Remote Creation Call (`RCC`). This tutorial covers the process of creating a Capsule over the network and give it a unique color.

Creating a Game script

Start off with a new project and import TNet into it.

Create a new GameObject in your scene and call it `GameManager`. Next create a C# script with the same name. Add the component script to the GameManager GameObject and then open the script in your script editor.

- Delete the default `Update` function.
- Replace `using System.Collections;` with `using TNet;` to the top of the file.

Connecting to a server

To connect to a server, we have to first start a server, and then connect to it.

```
void Start()
{
	// Start up a local server.
	TNServerInstance.Start("MyServer");

	// Connect to the server.
	TNManager.Connect();
}
```

Below the `Start()` function the code needs to first respond to the onNetworkConnect event and after that it will need to join the channel using onJoinChannel. OnEnable runs slightly before Start where the connect message is sent, so it's the best place to hook them up.

```
void OnEnable()
{
	TNManager.onConnect += OnNetworkConnect;
	TNManager.onJoinChannel += OnNetworkJoinChannel;
}

void OnDisable()
{
	TNManager.onConnect -= OnNetworkConnect;
	TNManager.onJoinChannel -= OnNetworkJoinChannel;
}

void OnNetworkConnect(bool success, string message)
{
	if (success)
	{
		TNManager.JoinChannel(1);
	}
	else Debug.LogError("OnNetworkConnect(): " + message);
}

void OnNetworkJoinChannel(int channelID, bool success, string message)
{
	if (success)
	{
		// We're now in a channel
	}
	else Debug.LogError("OnNetworkJoinChannel(): " + message);
}
```

Once the client is connected it needs to enter a channel, that's done in the function `OnNetworkConnect`. Channels are used to differentiate scenes or rooms to keep network traffic to a minimum. In this example everyone will join to channel 1.
Now that the client is connected and in a channel they can spawn the capsule and color it.

Spawning a GameObject with parameters

- Back in Unity place a capsule in the scene at (0, 0, 0).

- Add the `Network Object` script (TNObject) to it. Think of Network Object as a house number residing on a street with the name matching your channel ID.
- Next create a folder in the project window and name it `Resources`
- Drag the capsule from the Hierarchy window and into the Resources folder in the Project window to create a prefab of it.
- Delete the Capsule from your scene.

You should now have a prefab in the Resources folder that has a network object script attached to it.

In the `GameManager OnNetworkJoinChannel()` function add the following line to the success section.

```
if (success)
{
	// We're now in a channel
	Vector3 v = new Vector3(Random.Range(-3f, 3f), 0f, Random.Range(-3f, 3f));
	Color c = new Color(Random.value, Random.value, Random.value, 1f);


	TNManager.Instantiate("CreateWithColor", "Capsule", false, v, c);
}
```

The first parameter of the Instantiate command is the name of the RCC function that will be called to handle the creation of the object. Next is the path and name of the prefab located in the Resources folder. “false” tells the software that the object should not be persistent. This means that once the user disconnects from the server the object will be destroyed. It’s useful for things like player avatars. If your game’s user is creating long lasting objects, such as a house, then persistent would be set to true.

Now the CreateWithColor function needs to be properly defined. Create a new static function which returns a GameObject called CreateWithColor and mark it with the [RCC] attribute.

```
[RCC]
static GameObject CreateWithColor(GameObject go, Vector3 pos, Color c)
{
	go = Instantiate(go, pos, Quaternion.identity) as GameObject;
	go.GetComponent<Renderer>().material.color = c;
	return go;
}
```

This function is called on every client and will locally instantiate the specified prefab being sent in. Using the second and third parameters, it positions the object and then applies the specified colour to the Renderer.

Press play and you will see a yellow capsule appear in the world.