6.1 Project Optimization

Techniques:

1: Variable attributes

2: Unity Event Functions

3: Object Pooling

Length:	30 minutes
Overview:	In this lesson, you will learn about a variety of different techniques to optimize your projects and make them more performant. You may not notice a huge difference in these small prototype projects, but when you're exporting a larger project, especially one for mobile or web, every bit of performance improvement is critical.
Project Outcome:	Several of your prototype projects will have improved optimization, serving as examples for you to implement in your personal projects
Learning Objectives:	By the end of this lesson, you will be able to: - Recognize and use new variable attributes to keep values private, but still editable in the inspector - Use the appropriate Unity Event Functions (e.g. Update vs. FixedUpdate vs. LateUpdate) to make your project run as smoothly as possible - Understand the concept of Object Pooling, and appreciate when it can be used to optimize your project

1: Variable attributes

In the course, we only ever used "public" or "private" variables, but there are a lot of other variable attributes you should be familiar with.

1. Open your **Prototype 1** project and open the **PlayerController.cs** script
2. Replace the keyword "private" with **[SerializeField]**, then edit the values in the inspector
3. In **FollowPlayer.cs**, add the **[SerializeField]** attribute to the Vector3 **offset** variable
4. Try applying the "**readonly**", "**const**", or "**static**" attributes, noticing that all have the effect of removing the variable from the inspector

- **New Concept:** using [SerializeField] instead of public attribute
- **Tip:** "protected" is very similar to "private", but would also allow access to derived classes

```
[SerializeField] ~~private~~ float speed = 30.0f;
[SerializeField] ~~private~~ float turnSpeed = 50.0f;

[SerializeField] ~~private~~ Vector3 offset = new Vector3(0, 5, -7);
```

2: Unity Event Functions

In the course we only ever used the default Update() and Start() event functions, but there are others you might want to use in different circumstances

1. **Duplicate** your main Camera, rename it "Secondary Camera", then **deactivate** the Main Camera
2. **Reposition** the Secondary camera in a first-person view, then edit the **offset variable** to match that position
3. Run your project and notice how choppy it is
4. In **PlayerController**.cs, change "Update" to "FixedUpdate", and in **FollowPlayer**.cs, change "Update" to "LateUpdate", then **test again**
5. **Delete** the Start() function in both scripts, then reactivate your Main Camera

- **New Concept:** "Event Functions" are Unity's default methods that run in a very particular order over the life of a script (e.g. Start and Update)
- **New Concept:** Update vs FixedUpdate vs LateUpdate
- **New Concept:** Awake vs Start
- **Tip:** If you're not using Start or Update, it's cleaner to delete them

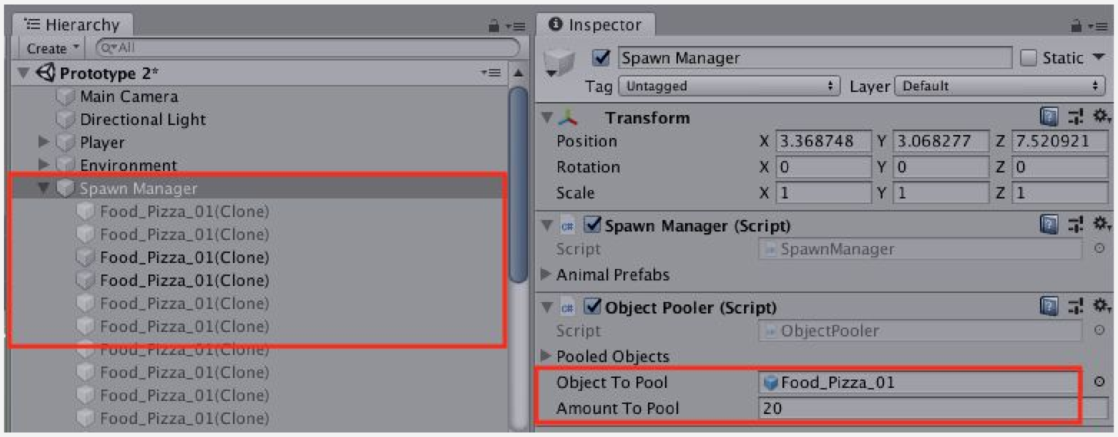
```
PlayerController.cs
void FixedUpdate() { ...

FollowPlayer.cs
void LateUpdate() { ...
```

3: Object Pooling

Throughout the course, we've created a lot of prototypes that instantiated and destroyed objects during gameplay, but there's actually a more performant / efficient way to do that called Object Pooling.

1. Open **Prototype 2** and create a backup
2. **Download** the **Object Pooling** unity package and **import** it into your scene
3. Reattach the **PlayerController** script to your player and reattach the **DetectCollisions** script to your animal prefabs (***not*** to your food prefab)
4. Attach the **ObjectPooler** script to your Spawn Manager, drag your projectile into the "**Objects To Pool**" variable, and set the "**Amount To Pool**" to 20
5. **Run** your project and see how the projectiles are activated and deactivated

- **Warning**: You will be overwriting your old work with this new system, so it's important to make a backup first in case you want to revert back
- **New Concept**: Object Pooling: creating a reusable "pool" of objects that can be activated and deactivated rather than instantiated and destroyed, which is much more performant
- **Tip**: Try reading through the new code in the ObjectPooler and PlayerController scripts
- **Don't worry**: If your project is small enough that you're not experiencing any performance issues, you probably don't have to implement this



Spawn Manager
Tag Untagged Layer Default
Transform
Position X 3.368748 Y 3.068277 Z 7.520921
Rotation X 0 Y 0 Z 0
Scale X 1 Y 1 Z 1
Spawn Manager (Script)
Script SpawnManager
Animal Prefabs
Object Pooler (Script)
Script ObjectPooler
Pooled Objects
Object To Pool Food_Pizza_01
Amount To Pool 20

Lesson Recap

New Concepts and Skills

- Optimization
- Serialized Fields
- readonly / const / static / protected
- Event Functions
- FixedUpdate() vs. Update() vs. LateUpdate()
- Awake() vs. Start()
- Object Pooling