FIMPOSSIBLE
CREATIONS

OPTIMIZERS

USER MANUAL

About Optimizers

- Optimizers are components implementing Unity Culling Groups API and using other techniques to check objects state in relation with camera's distance and visibility.

- With data of object relation with the game camera optimizers are applying different settings at selected distances for any type of component (particle systems / lights / MonoBehaviours, Terrains etc.)

- When using optimizers with Culling Groups Containers (spheric detection) you can use even 100 000 of optimizers without much performance load on checking objects state if use multiple objects with same LOD levels count and same distance values.

- You should not use optimizers just on single mesh instances for culling, for that build in LOD Groups are better, but still you can use Optimizers anyway.
When using 100 000 of LOD groups it can take about 1.4ms to 2.4ms for camera culling, when using 100 000 contained Optimizers it can take from 0.5ms up to 1.6ms so difference will be a bit noticeable only in very big numbers of game objects.

- Optimizers are giving the biggest performance boost on optimizing particles.
If you have many particle effects then Optimizers can do a lot.
If you do not use as many particles then now with optimizers you can use more of them!

- Package is providing tools to convert old Optimizers V1.0 scripts to V2, it can be unpacked to project with additional .unitypackage

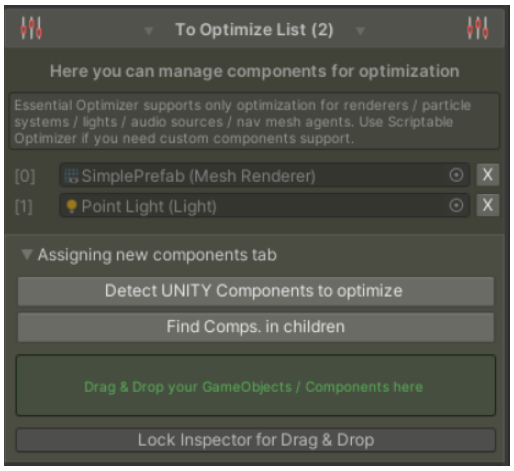
Contact and other links you will find in Readme.txt file

Index

1: Getting Started

Setup:

Add **Essential Optimizer** to your object.
Component will automatically try to find Unity Components for optimization if some will be found then you will see them under "To Optimize List" and you can remove ones you don't want to change.


To Optimize List (2)

Here you can manage components for optimization

Essential Optimizer supports only optimization for renderers / particle systems / lights / audio sources / nav mesh agents. Use Scriptable Optimizer if you need custom components support.

[0] SimplePrefab (Mesh Renderer) X
[1] Point Light (Light) X

Assigning new components tab

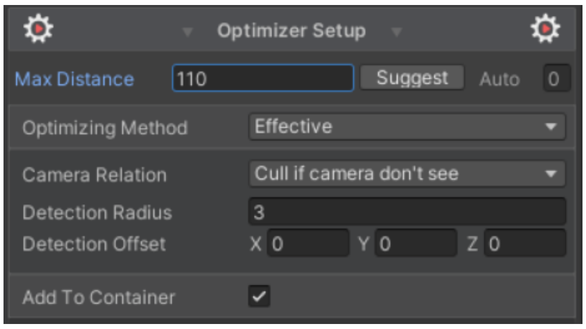
Detect UNITY Components to optimize

Find Comps. in children

Drag & Drop your GameObjects / Components here

Lock Inspector for Drag & Drop

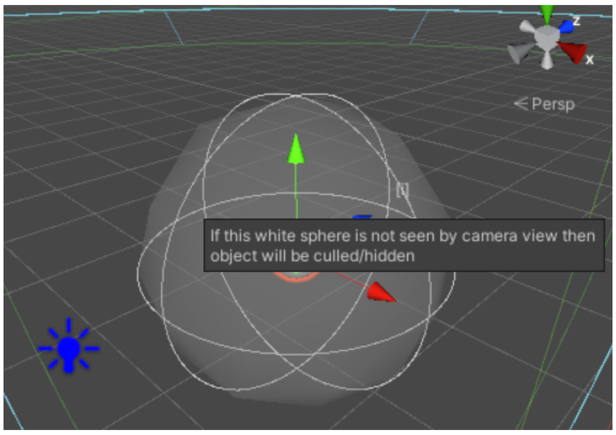
You can try searching for other components in all child transforms of game object or try finding custom MonoBehaviours for optimization with buttons or by drag & dropping game objects from the scene.


Optimizer Setup

Max Distance	110 Suggest Auto 0
Optimizing Method	Effective
Camera Relation	Cull if camera don't see
Detection Radius	3
Detection Offset	X 0 Y 0 Z 0
Add To Container	✓

Define the farthest distance for your components to optimize group.
After reaching this distance all components will be disabled or other custom action will be applied to them (to customize in LOD levels)

You can disable or change components quality settings **when the camera looks away from the object**.



If this white sphere is not seen by camera view then object will be culled/hidden

With **Detection Radius** and **Detection Offset** you will adjust how camera should detect your components group visibility. All of this is visualized on **scene view**.

Optimizing Methods:

Optimizers are providing four different optimization methods.
Default and recommended one is called **Effective**.
Effective optimizing method is using Unity Culling Group API, Optimizer's Dynamic Clock and Optimizer's Culling Containers.
This combination is providing very performant and effective object state detection without frame-skips.
Static method is using the same techniques but it don't need to use Dynamic Clock since static object are not changing position.
Objects which are using this two methods (Effective or Static) are able to be put inside **Culling Container** which will be described later.

Dynamic method is using just Dynamic Clock to update object visibility state. Since it is not using CullingGroups API detection shape is box.
(I recommend using Effective method even box is fitting better to your game model)
Visibility state is updated every frame if object is near, if further then update rate is lower, update rate depends on cpu load of clock, if too many objects then update rate is also slowing down.
Distance ranges and update rate for Dynamic Clock can be adjusted with Optimizers Manager Component.

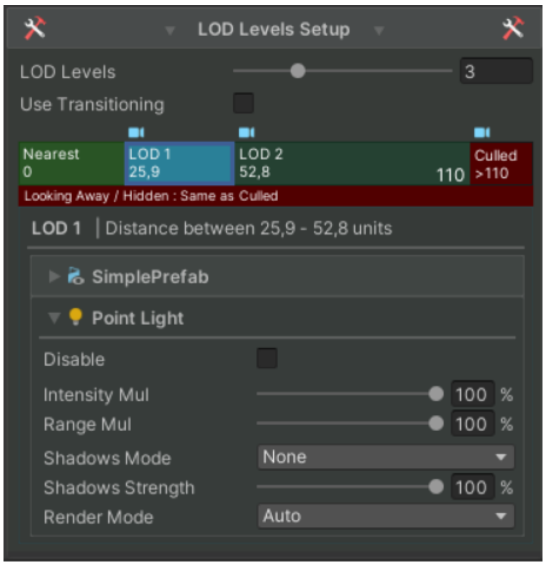
Trigger Based is using trigger colliders for detecting distance ranges and for camera visibility is using dynamic clock. It's experimental and not tested enough for final products usage.

If your game is changing Render Camera dynamically you can help optimizers identify correct camera by calling:

```
FIMSpace.FOptimizing.OptimizersManager.SetNewMainCamera(YourCamera);
```

when you change your camera during gameplay through code.

LOD Levels Settings:


LOD Levels Setup

LOD Levels 3

Use Transitioning

Nearest 0 | LOD 1 25,9 | LOD 2 52,8 | 110 | Culled >110

Looking Away / Hidden : Same as Culled

LOD 1 | Distance between 25,9 - 52,8 units

SimplePrefab

Point Light

Disable

Intensity Mul 100 %

Range Mul 100 %

Shadows Mode None

Shadows Strength 100 %

Render Mode Auto

You can define how many levels of quality steps should have your components group. In each level you can customize what settings should be applied to your components.

You can enable transitioning to fade parameters smoothly when LOD levels are changing. (some parameters can't be transitioned)

You can define different settings than culling **when the camera is near** to your object **and look away**. It's useful when for example you have an AI movement script (and a bunch of other components like particles etc.) which you want to disable when the camera is far but when camera is just looking away from the object it shouldn't be culled and disabled. You can select Hidden Settings bar and define from which LOD level looking away should cull object and define other settings when camera is near and looking away (it's called Hidden LOD Settings).

By clicking on blue camera icons, the scene view camera will move to LOD range position, so you can check on what zoom levels object will change target LOD level or be culled.

Optimizing MonoBehaviours:

Some game scripts can take a lot of CPU if there are some of them used on scene.
(Even simple transform.Rotate scripts if there is a lot of them on scene)
Maybe you will find some scripts which don't need to be executed when the camera is far from them or not looking at objects which are using these scripts.

It can be scripts which are animating objects through script, AI scripts or maybe you can find other ones in your project. Feel free to experiment with optimizing custom MonoBehaviour scripts.

2: Correct Workflow

Culling Containers:

To take the best of Optimizers you need to know for what Culling Containers are and how to manage optimizers to use their potential.

Culling Containers are Unity's Culling Groups (multiple if needed) but dynamically managed with capacity control.

When you add optimizer to some object and adjust MaxDistance with LOD levels count make sure to not change MaxDistance value or LOD sliders for single LOD Level on each copy of object with optimizer.

When optimizers are using **same LOD levels count**, **same MaxDistance** and **same Distance Ranges** for LOD levels then you can compute visibility state **for like thousand copies** of them by cpu use of computation **for single one optimizer**.

Only thanks to Culling Containers you can use 100 000 optimizers without hurting the CPU and still be very efficient with visibility states.

So remember to use the same distance parameters on different optimizers if of course you can allow for that.
If you set different ranges for different types of models then other culling containers will be generated for each type of model. (or prefab?)

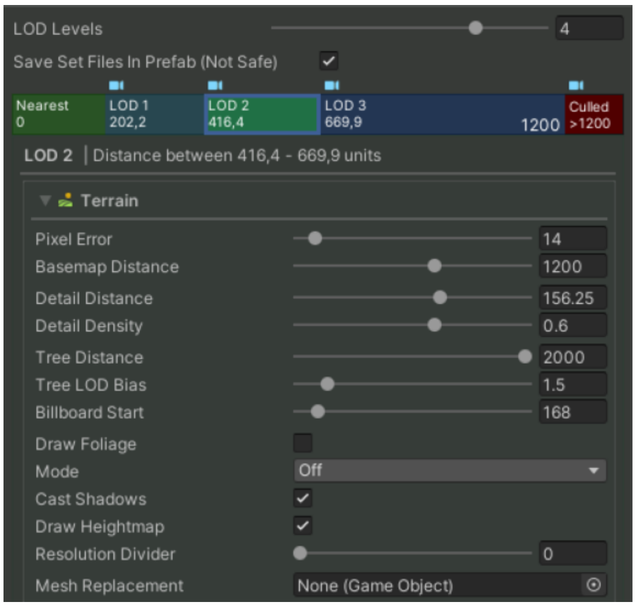
Try paying attention to it to avoid auto-creating unnecessary Culling Containers.
You can watch what is going on with Culling Containers through the **Optimizers Manager** component under **Containers Debug** tab. In this component you can also define how much optimizers slots should be prepared per Culling Container with **Single Container Capacity** parameter.
If you will use a lot of optimized objects make this value higher, if you will use only a few optimized objects you can make this value a bit lower.

If you use a lot of optimized objects and capacity per culling container is bigger then there will be generated smaller count of culling containers and performance will be slightly better. If you would use a lot of optimized objects with small capacity then many culling containers will be created and culling performance will cost a bit more.If you will use just a few optimized objects with very large capacity value then unnecessary data will be allocated so you will use tiny bit too much RAM than you could.

Terrain Optimizer:

Optimizers package is providing **Terrain Optimizer** which is using Culling Groups API to compute visibility state of the whole terrain component mesh.
<u>If you use just one big terrain in your game then Terrain Optimizer will not help you.</u>
If you use multiple terrains in your game then Terrain Optimizer can give you some additional FPS.

With Terrain Optimizer you have access to different terrain parameters which you can modify when the camera is going far from the terrain.



LOD Levels 4
Save Set Files In Prefab (Not Safe) ✓
Nearest 0 | LOD 1 202,2 | LOD 2 416,4 | LOD 3 669,9 | 1200 | Culled >1200
LOD 2 | Distance between 416,4 - 669,9 units

Terrain

Pixel Error	14
Basemap Distance	1200
Detail Distance	156.25
Detail Density	0.6
Tree Distance	2000
Tree LOD Bias	1.5
Billboard Start	168
Draw Foliage	
Mode	Off
Cast Shadows	✓
Draw Heightmap	✓
Resolution Divider	0
Mesh Replacement	None (Game Object)

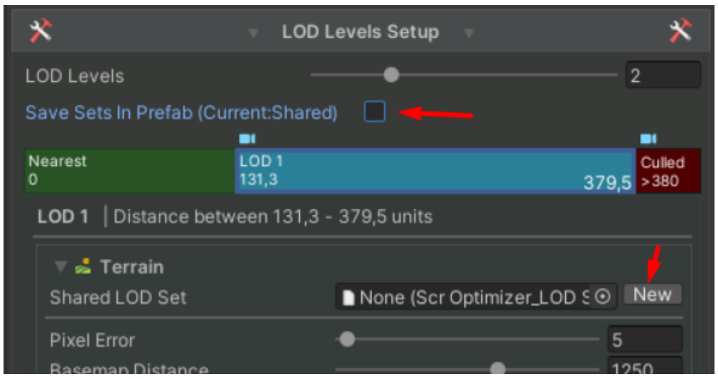
You can disable completely foliage when camera is looking away from terrain.
You can disable whole component drawing completely and use replacement mesh instead of terrain component (if you have prepared mesh).

Essential or Scriptable Optimizer:

<u>Essential Optimizer is recommended to use</u>. It serializes LODs data with component instead of scriptable files like **Scriptable Optimizer** does.

Scriptable Optimizer is for advanced users:
Each scriptable LOD file must be saved in the project directory in order to be remembered by anything in other scenes.
Each optimized component with Scriptable Optimizer needs to have saved LOD data and it can be annoying to every time create a new file for every optimized component.
To prevent that, Optimizers are trying to save mentioned files as Sub-Assets for prefabs when creating prefab.

With new Unity Prefabs Pipeline saving this Sub-Assets is broken and can't be programmed to be stable enough with optimizers requirements.
To use Scriptables and be stable you can disable **Save Set Files In Prefab** parameter and save LOD files by hand inside project like said before. It can seem heavy but with saved LOD files you can share settings between different prefabs and sometimes it can work better than using Essential Optimizer, it's up to you how you will use it.



LOD Levels Setup
LOD Levels 2
Save Sets In Prefab (Current:Shared)
Nearest 0 | LOD 1 131,3 | 379,5 | Culled >380
LOD 1 | Distance between 131,3 - 379,5 units
Terrain
Shared LOD Set None (Scr Optimizer_LOD S New
Pixel Error 5
Basemap Distance 1250

Scriptable Optimizer allows us to create custom implementations of LOD components to optimize which Essential Optimizer can't offer.
(You still can optimize custom MonoBehaviours with Essential Optimizer just in less comfortable way)
You must remember to set the same LOD levels count on your Scriptable Optimizer to be the same as other shared settings file to avoid conflicts. If you will anyway use **Save Set Files In Prefab** then always remember to apply override changes on prefabs and avoid changing lod levels through Prefab Mode (Isolated Scene) or through Scene Prefab, the best way is modifying prefab through project file selection.

When using **Save Set Files In Prefab** you should sometimes check prefab for sub-assets leftovers with Optimizers Cleaner component, also checking through project file selection on prefab, Unity can't get path to project prefab through isolated scene / scene prefab to clean sub-assets.

Another benefit from using Scriptable Optimizer is less RAM usage than Essential Optimizer, but the difference is very very small.

You can find additional components implementations for Optimizers under this link: Google Drive Folder after importing selected type components will be detected by scriptable object when searching for components to optimize.

You can check **this tutorial** for additional explanation on how to use scriptable shared settings with optimizers.

3: Additional Modules

Obstacles Detection:

With obstacles detection you can allow the optimizer to apply hidden LOD settings when objects are covered by some obstacles.
It's similar to occlusion culling but obstacle detection should not be used on thousands of single meshes, but rather on groups of meshes + other components.
Obstacles detection is using simple raycasting to detect collisions so if goal is hiding simple mesh object then raycast can cost more than drawing this mesh.
Use it feature wisely.
(upcoming DOTS implementation will give this feature new possibilities)
Please check **this tutorial video** for more precise explanation how it works and how to set it up properly.

Complex Detection Shape:

Complex Detection Shape allows you to define visibility shape by multiple spheres.
It can be helpful when a single sphere can't describe your model's shape.
(Even spheres will not fit perfectly it's recommended to use this instead of dynamic box shape)
Using **Culling Containers** with **Complex Detection Shape** needs a bit more operations to make it work and it's supported in the new version but needs to be tested more.

Warning:

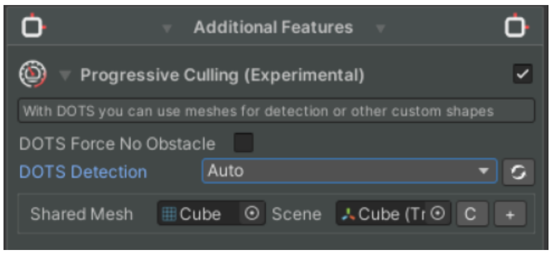
Complex Detection Shape can't be used with Obstacle Detection feature but it might be possible in next updates for optimizers.

Progressive Culling (Experimental)**:**

Progressive Culling can give results like Unity's Occlusion Culling but not only on static objects. This feature needs Unity 2018.4+ and imported packages (through Package Manager) like *Jobs, Burst, Mathematics and Collections*.
After importing these packages, new options on inspector GUI will pop up.

In comparison to Obstacle Detection, Progressive Culling is better when you have tones of objects to cull. When you have few objects which you want to cull behind other objects it's better to use obstacle detection instead of progressive culling.

You can define if a culled object should be detected by an optimized object's mesh model or Sphere/Cube shape, or you can create colliders and add to them "OptimizersReference" component to use them as culling detection shapes with "Custom" option.
In most cases you will probably use mesh, so "Auto" option.


Additional Features

Progressive Culling (Experimental) ✓

With DOTS you can use meshes for detection or other custom shapes

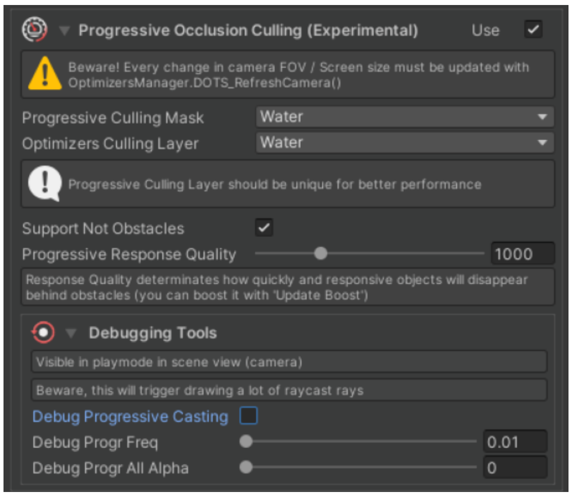
DOTS Force No Obstacle ☐

DOTS Detection: Auto

Shared Mesh: Cube | Scene: Cube (Tr | C | +

If you want to use Progressive Culling on transparent objects like Light, you should mark them as "No Obstacle", so Light's culling detection shape can be covered by other objects and hidden, but Light's detection shape will not cover other objects - like it's transparent. (If progressive culling system detect just light to optimize it will make it force "No Obstacle" automatically, so you don't need to bother)

Progressive Culling is not hiding objects instantly.
When an object is covered by another object or is out of camera view it needs a few seconds to be hidden. Progressive Culling is casting a lot of collision rays but not all at once, it's spreading them over frames with fast "Jobs" compilation (async multithreading) to make the system work smoothly.
(If objects would be hidden instantly when any ray did not hit it, then objects around it would flicker, being enabled and disabled again every few frames.)


Progressive Occlusion Culling (Experimental) Use ✓

Beware! Every change in camera FOV / Screen size must be updated with OptimizersManager.DOTS_RefreshCamera()

Progressive Culling Mask: Water

Optimizers Culling Layer: Water

Progressive Culling Layer should be unique for better performance

Support Not Obstacles ✓

Progressive Response Quality: 1000

Response Quality determinates how quickly and responsive objects will disappear behind obstacles (you can boost it with 'Update Boost')

Debugging Tools

Visible in playmode in scene view (camera)

Beware, this will trigger drawing a lot of raycast rays

Debug Progressive Casting ☐

Debug Progr Freq: 0.01

Debug Progr All Alpha: 0

Inside the Optimizers Manager you will find a few additional parameters for adjusting Progressive Culling performance and precision or debug it.

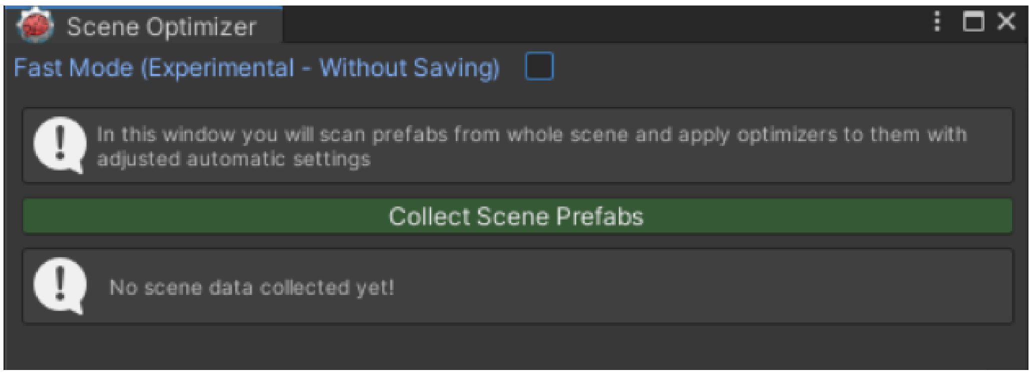
If you set "Progressive Response Quality" higher then objects will be hidden quicker and more rays will be casted for bigger precision.

Scene Tools (Experimental)**:**

With Optimizers version 2.1.5 comes a tool window to collect all prefabs from the active editor scene, define size and apply optimizers to them with according settings.

To access this window open Optimizers Manager -> Scene Tools -> Open Scene Optimizer Window

You should find new window looking like this:



In this window you will scan prefabs from whole scene and apply optimizers to them with adjusted automatic settings

No scene data collected yet!

Since Unity provided a new Prefabs Pipeline it's extremely slow, if you have a very big amount of prefabs on the scene, unity's asset database will consume a lot of time, please be patient. (In Unity 2018.3 and lower this process is a lots quicker)
However you can partially avoid it toggling "Fast Mode" but after Scene Optimizer work, hitting "Ctrl+s" will probably trigger Asset Database to go through every modified prefab anyway.

Hitting "Collect Scene Prefabs" will make Scene Optimizer search scene for all Prefabed objects. After that you can define parameters for optimizers like max culling distance for small objects and max culling distance for big objects.
Manager will categorize these distances into groups on which you can customize some settings with "Customize Group Parameters" foldout.

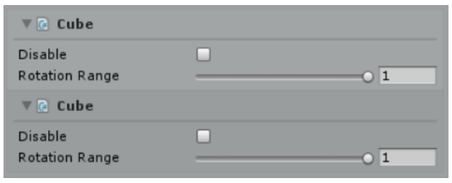
Below you have a panel to preview desired automatic scaling for every prefab and do adjustment if needed.

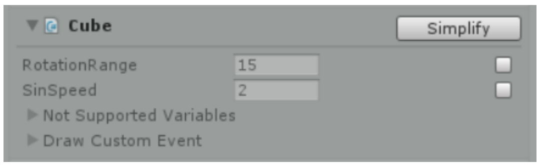
Hitting "Add Optimizers To Prefabs" will add Essential Optimizer to every prefab.
You can update settings on every optimizer on the scene with the "Adjust Optimizers" tab on the top of the Scene Optimizer Window.

Tool will have more options in next updates.

4: Coding Custom LOD Type Instance

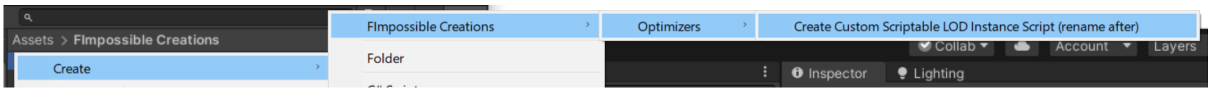
This feature is available with Scriptable Optimizer.
You can create a custom LOD menu for your component type.
When you add MonoBehaviour to optimize then the optimizer is trying to draw all it's parameters, when for example you need to use only two parameters and set them automatically you can create custom LOD Type Instance **or just use shared settings.**





Custom LOD Instance Instance **VS** **MonoBehaviour LOD**

To create a custom LOD Type Instance you can use a script template.
Click with right mouse button somewhere inside project then select:


Assets > FImpossible Creations
Create
FImpossible Creations
Folder
Optimizers
Create Custom Scriptable LOD Instance Script (rename after)
Collab
Account
Layers
Inspector
Lighting

New script file will appear, **you should name it like your component**
(you need to rename this file later) thanks to that template will be filled with your component type name automatically.

Now you have to define how LOD Instance will work.
This tutorial can be helpful, it's for Optimizers Version 1 but still very relatable.

After filling code with your parameters you have to create a reference asset.
Inside "Scriptable Container for IFLOD Instance" region there is line "[CreateAssetMenu(..." which is creating new menu when you hit right mouse button in the project **Fimpossible Creations -> Optimizers -> Custom Optimizers -> ScrLOD_YOURCOMPONENT Reference**.
Reference .asset will be created and you have to move it to "Optimizers 2/Resources/Custom"
Last thing will be renaming your custom component's LOD Instance.cs script file with the same name as the generated class name below line "[CreateAssetMenu(..." line so: ScrLOD_**YOURCOMPONENT**.

When everything is done Optimizers will try to find component of your type when searching for new ones and generate LOD Instance of your type instead of MonoBehaviour.

Optimizer 2019.4+ / “Optimizer 2”:

This component is available only in Unity 2019.4+ versions.
It’s using the new feature “Serialize Reference” to avoid using scriptable objects.
There is very simple example package how to implement custom component support, similar to how a scriptable optimizer is handling that.
If you don’t need to use custom components setup I recommend sticking to the Essential Optimizer since the Optimizer 2019.4+ needs more testing to ensure full stability on every platform.

Disable auto-gathering of some component types:

If there are components you don’t want to be gathered automatically by Optimizers when being added to the object, you can go to the “EssentialSelector” or “Optimizers2020Selector” file and choose components you want/don’t want to be included by optimizers (drag & drop option will still include the untoggled components)

5: Optimizers V1 Converter to V2

If you was using Optimizers V1 you need to convert old components to new ones in order to use new features.

In Optimizers Version 2 LOD setting etc. are built on different structure and settings from V1 can’t be used on Optimizers V2.

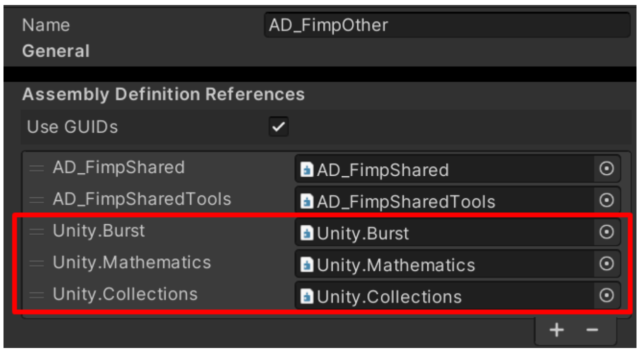
Optimizers V2 provides .unitypackage with Optimizers V1 with additional UI for converting old component’s settings to new one.

It’s recommended to use converter on prefabs which through project browser window, no through Prefab Isolated Mode or Prefab placed on scene.
It will be more clean for Optimizers to remove Optimizers 1 correlations.

6: Additional Info

Progressive Culling:
Progressive Culling feature requires few additional packages by unity:
mathematics, jobs, collections and burst.
If you use some newer version of the unity, this packages may not be yet officially updated up
to the newest unity version, but you can try importing them manually by modifying packages
manifest file or using "+" button in package manager and selecting "Add package by name"
then you need to provide packages like:
com.unity.mathematics   *com.unity.jobs*   *com.unity.collections*   *com.unity.burst*

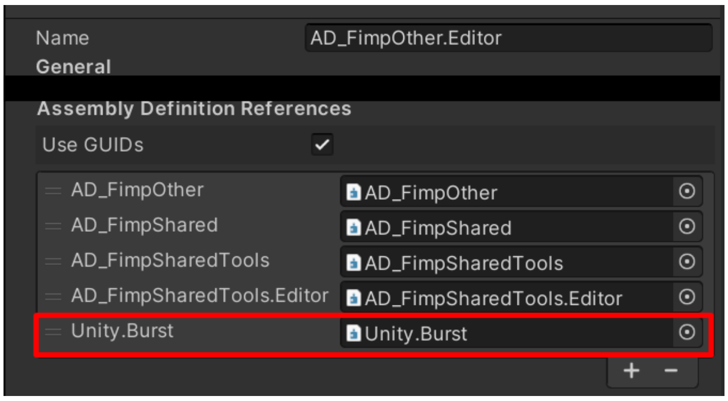
Assembly Definitions and Progressive Culling:
When using assembly definitions and jobs/burst extension, you will need to add few references to the assembly definition file **'AD_FimpOther'** which are:
Unity.Burst   *Unity.Collections*   *Unity.Mathematics*



Name AD_FimpOther
General
Assembly Definition References
Use GUIDs ✓

AD_FimpShared	AD_FimpShared
AD_FimpSharedTools	AD_FimpSharedTools
Unity.Burst	Unity.Burst
Unity.Mathematics	Unity.Mathematics
Unity.Collections	Unity.Collections

Similar thing for **'AD_FimpOther.Editor'**:
Unity.Burst   *Unity.Collections*



Name AD_FimpOther.Editor
General
Assembly Definition References
Use GUIDs ✓

AD_FimpOther	AD_FimpOther
AD_FimpShared	AD_FimpShared
AD_FimpSharedTools	AD_FimpSharedTools
AD_FimpSharedTools.Editor	AD_FimpSharedTools.Editor
Unity.Burst	Unity.Burst

If the project is still not compiling, try closing unity editor and run it again.

Removing burst/jobs support from the project:
If you removed the packages listed above, optimizers should automatically take care for hiding
classes which requires the packages, but if not, you can go to:
Project Settings -> Player -> Other Settings -> Script Compilation
and remove 'OPTIMIZERS_DOTS_IMPORTED' if it's still here: it will hide
content of some classes which are using burst/jobs methods.