Impossible Odds - Tactical Camera

The Impossible Odds Tactical Camera package provides a plugin camera system for smoothly navigating your environments in both a top-down tactical view as well as up-close action scenes.

You can expect to find the following set of features in this plugin:

- Move the camera using the keyboard or screen edge detected, or double click to move to a target position.
- Zoom in & out with a dynamic field of view to get a greater sense of scale.
- Look around with restricted tilt angles, and orbit around a focus point.
- Smooth collision detection with terrain and objects in your world.
- Restricted area-of-operation to keep the camera inside the map boundaries.
- Height-based parameters where the camera's behaviour changes based on its altitude.
- Extensive customization of behaviour through simple animation curves.
- Minimal setup and easy integration.

You can view a quick demo video of this plugin below.

Quick Camera Setup

If you want to get started quickly without being bothered with all the details, then follow these few steps:

1. Add the `TacticalCamera` component to your `Camera` GameObject. This will also add a `CharacterController` component to it.
2. Apply settings by picking a `TacticalCameraSettings` data object from your project. You can play around with the different values to make it feel/behave differently. The package comes with one already defined for you to use.
3. Finally, add the `TacticalCameraInputProvider` component to it and assign it to the `TacticalCamera` component. Feel free to adjust the key bindings in a way that you see fit.
4. Optionally, add some bounds in which the camera is allowed to operate and assign it to the `TacticalCamera` component. The `TacticalCameraBoxBounds` component restricts it to an *axis-aligned* box.

That's it!

Inspector	
Main Camera	Static
Tag MainCamera	Layer Default
Transform	
Camera	
Audio Listener	
Character Controller	
Tactical Camera (Script)	
Script	TacticalCamera
Initial Settings	TacticalCameraSettings (TacticalCameraSettings)
Initial Input Provider	Main Camera (TacticalCameraInputProvider)
Initial Camera Bounds	Main Camera (TacticalCameraBoxBounds)
Tactical Camera Input Provider (Script)	
Tactical Camera Box Bounds (Script)	

Note: remember to check out the tooltips in the inspector if something is not clear!

Detailed Camera Setup

The Tactical Camera plugin has a central component that does all the heavy lifting: `TacticalCamera`. It will also add a `CharacterController` component on there which it uses to detect collisions as well as move smoothly over any terrain and objects.

The component requires a few additional data objects to operate correctly:

- An input provider to move and rotate in and around the environment, and
- A settings object that defines how it operates/behaves, e.g. movement and rotation speed, tilt angles, etc.

Optionally, the camera can also be equipped with a component that restricts its area of operation so that it can't move outside of your game world.

Tactical Camera (Script)	
Script	TacticalCamera
Initial Settings	None (Tactical Camera Settings)
Initial Input Provider	None (Abstract Tactical Camera Input Provider)
Initial Camera Bounds	None (Abstract Tactical Camera Bounds)

Input Provider

The input provider for the Tactical Camera instructs where the camera should go to and where to look at. Input is of course very project-dependent and there is no "one size fits all"-solution, e.g. mouse & keyboard versus gamepad versus touch screen, etc. That's why this camera is designed to be driven using an interface-component rather than forcing a single solution on you. However, a sample implementation of an input provider is given by the `TacticalCameraInputProvider` component. It's designed for use with mouse and keyboard and allows you to assign which keys and axes should be used to control the movement and rotation.

Movement	
Move Forward Key	W
Move Backward Key	S
Move Left Key	A
Move Right Key	D
Orbit Camera	Left Shift
Screen Border Trigger	0
Trigger Off Screen	☐
Scroll Sensitivity Factor	100
Rotation	
Mouse Move To Position Key	Left
Mouse Rotation Key	Right
Mouse Tilt Axis	Mouse Y
Mouse Rotation Axis	Mouse X
Rotation Modifiers	
Invert Tilt	☑
Invert Rotation	☐
Invert Zoom	☑
Always Rotating	☐

Note: remember to check out the tooltips in the inspector if something is not clear!

Input Provider - Advanced

The `ITacticalCameraInputProvider` interface is the key to hooking the Tactical Camera system in your custom input code. Either have your input object implement this interface as well to pass it along to the Tactical Camera directly, or provide an abstraction layer for this interface in your input code.

The interface expects you to implement the following features:

- Moving forward, sideways and upwards through the `MoveForward`, `MoveSideways` and `MoveUp` properties. These should return a value in the range of [-1, 1].
- Moving to a target position through the `MoveToTarget` and `CancelMoveToTarget` properties. These should return `true` when the action is requested, i.e. when is button is pressed down that frame.
- Rotating around its pivot or focus point through the `TiltDelta` and `RotationDelta` properties. The `OrbitAroundTarget` is the switch between rotating around the pivot or focus point. As long as this last one keeps returning `true`, the Tactical Camera will orbit rather than rotate around its pivot.

Note: the `ITacticalCameraInputProvider` interface in itself does not define which particular input method is used, e.g. moving using the keyboard versus the mouse cursor against the edge of the screen. The implementing class should define which methods are supported by way of returning appropriate values.

Settings

The Tactical Camera's behaviour is driven by a data object that tells it how fast it can move and rotate, as well as how smooth it comes to a standstill and how its movement is affected by altitude. The `TacticalCameraSettings` is the ScriptableObject that holds this data. It can be created through the Unity create menu: `Assets → Create → Impossible Odds → Tactical Camera → new Tactical Camera Settings`.

You can adjust the values in this object to suit the environment the camera will operate in, but sensible default values have already been set for a small-to-medium sized map. This object operates using value ranges and animation curves. The choice for using animation curves allows to visually define how a specific value should fade out, or how a value should transition from one end to the other, e.g. the movement speed versus the altitude of the camera.

The settings object also allows to set some world-interaction values which might be of particular interest for performance reasons:

- The `InteractionMask` property defines which collision layers are interacted with during raycast operations. This is best set to only layers that are of interest to the camera to interact with, i.e. your terrain and world-objects.
- The `InteractionDistance` property defines the maximum length of raycasts used for interacting with your world. The larger this value, the further it has to check. Reduce this to a value that makes sense for your world setup.
- The `InteractionBubbleRadius` defines the sphere in which the camera collides with objects in your world.

Movement Settings	
Movement Speed Range	
Movement Speed Transition	
Movement Fade	
Movement Fade Time	0.2
Move To Target Smoothing Time	0.2
Absolute Height Range	
Rotation Settings	
Max Rotation Speed	180
Rotational Fade	
Rotational Fade Time	0.2
Tilt Settings	
Tilt Range Low	
Tilt Range High	
Tilt Range Transition	
Field of View	
Use Dynamic Field Of View	☑
Dynamic Field Of View Range	
Dynamic Field Of View Transition	
World Interaction Settings	
Interaction Mask	Default
Interaction Distance	1000
Interaction Bubble Radius	1

Note: remember to check out the tooltips in the inspector if something is not clear!

Settings - Advanced

Compared to the input provision, this implementation of settings is immediately usable in a broad set of projects. Nonetheless, if you want to further alter or customize the behavior, the `ITacticalCameraSettings` interface is what you should look into.

This interface will ask you to implement the following:

- The height limits in which the Tactical Camera is allowed to operate in through the `AbsoluteHeightRange` property.
- Movement behaviour through `MovementSpeedRange`, `MovementFadeTime` and `MoveToTargetSmoothingTime`.
- Rotational behaviour through `MaxRotationalSpeed`, `RotationalFadeTime`, `TiltRangeLow` and `TiltRangeHigh`. These last two define what the angel ranges are for the Tactical Camera when its at its lowest and highest point, respectively.
- Field of view settings through `UseDynamicFieldOfView` and `DynamicFieldOfViewRange`.
- World interation settings through `InteractionMask`, `InteractionDistance` and `InteractionBubbleRadius`.
- Several evaluation methods for custom fade-outs and transition curves.

Most of these properties are self-explanatory, but the purpose of the of *evaluation* methods might not be immediately clear. Each of these methods take a `float` value as input in the range of [0, 1]. The output of these methods can be exactly the same as their input value in case a linear transition is desirable, but it can also be transformed to create a more pleasant looking/feeling result, e.g. a smooth roll-off in case of fading, or an ease-in, ease-out in case of transitions. In the implementation of `TacticalCameraSettings`, this is done by evaluating animation curves which naturally lend themselves to this.

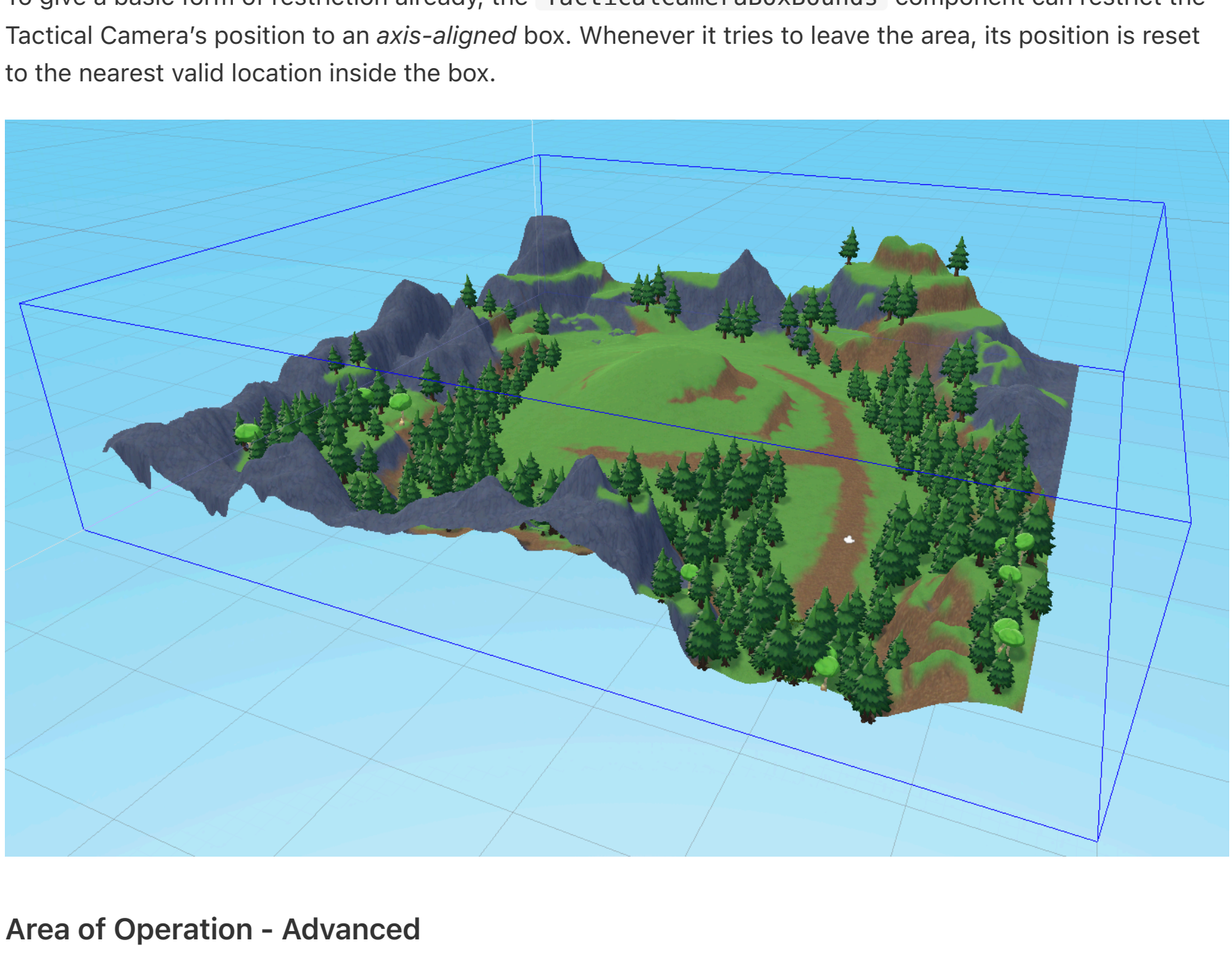
- `EvaluateMovementFadeOut` and `EvaluateRotationFadeOut` are fading methods that determine how the movement and rotation of the camera fade out when no more input is given.
- `EvaluateMovementTransition`, `EvaluateTiltTransition` and `EvaluateFieldOfViewTransition` methods determine how the camera reacts to transitions in altitude. When the camera moves around, several values or value ranges may change in response to that, e.g. when higher up, the faster the camera can move around.

Area of Operation

In most situations it's desirable to restrict the camera to a specific area so that it can't wander off in areas not meant for the players to visit. The Tactical Camera can equipped with some bounds that force it to be in a particular area.

How such an area is defined is very project-dependent and could become a complex matter. For example, polygonal level bounds in case the map is not just a square, or a fog-of-war restriction where undiscovered pieces of the map may not be accessible yet.

To give a basic form of restriction already, the `TacticalCameraBoxBounds` component can restrict the Tactical Camera's position to an *axis-aligned* box. Whenever it tries to leave the area, its position is reset to the nearest valid location inside the box.



Area of Operation - Advanced

To have the Tactical Camera be restricted to more complex areas, have your restriction tool implement the `ITacticalCameraBounds` interface. This has a single `Apply` method which will get called by the Tactical Camera once it has done moving.

Your implementation of this `Apply` method should check whether the camera is still in a valid location, and if not, place it somewhere that is. Preferably this would be the closest valid point inside to minimise the distance which would otherwise create weird visual jumps.

Dependency Injection

The external resources the Tactical Camera system requires to operate can all be assigned through Unity's inspector view or through your own scripts. Additionally, you can also have them delivered by the *Dependency Injection* framework from the Impossible Odds C# Toolkit. It allows you to inject resources and values into objects that require them to operate, which makes managing resources in your project easier.

The `TacticalCamera` component's properties for the input provider, settings and bounds are all marked for injection in case you prefer to work using this system as well.

Gotcha's

When using the Tactical Camera system you might run into a few limits of the system as well. You'll find them listed here:

- The `z`-value of the local Euler rotation angle is always set to 0 at the end of its `LateUpdate` phase. This is to prevent drift and keeps the camera straight up.
- The operating range of the tilt angle in the `TacticalCameraSettings` objects can be no larger than [-90, 90] degrees, with 0 degrees being level with the horizon. This range is defined to prevent flipping over. If you make a custom implementation through `ITacticalCameraSettings`, best is to keep this in mind.

Example

Check out the demo scene for a complete setup and demonstration of how the camera behaves in different circumstances. This scene is setup using the dependency injection method, i.e. the resources the camera needs are injected upon Start from the `TacticalCameraDemo` component.

Unity Version

Developed and tested on Unity 2019.4 LTS.

License