# Resolution Magic 2D Documentation


**Version 4.0**

**No black bars, no stretching…your game looking great on any screen.**

# Introduction

## What is Resolution Magic 2D?

Resolution Magic 2D is an add-on asset for the Unity game engine. It automates the hard work required to tailor games to the multitude of different screen ratios and resolutions available on modern devices, especially for platforms that have variable resolutions, like Android and Windows 8.

## Why do I need it?

Resolution Magic:

- Ensures all players see the same game content
- Adjusts to changes in orientation and resolution during gameplay
- Works on all platforms and doesn't interfere with other Unity features (e.g. Unity UI)

NO MORE:

- black bars (unless you want them – they are included as an optional feature)
- stretching
- fiddling around to test on different resolutions.

## That does sound like magic…how does it actually work?

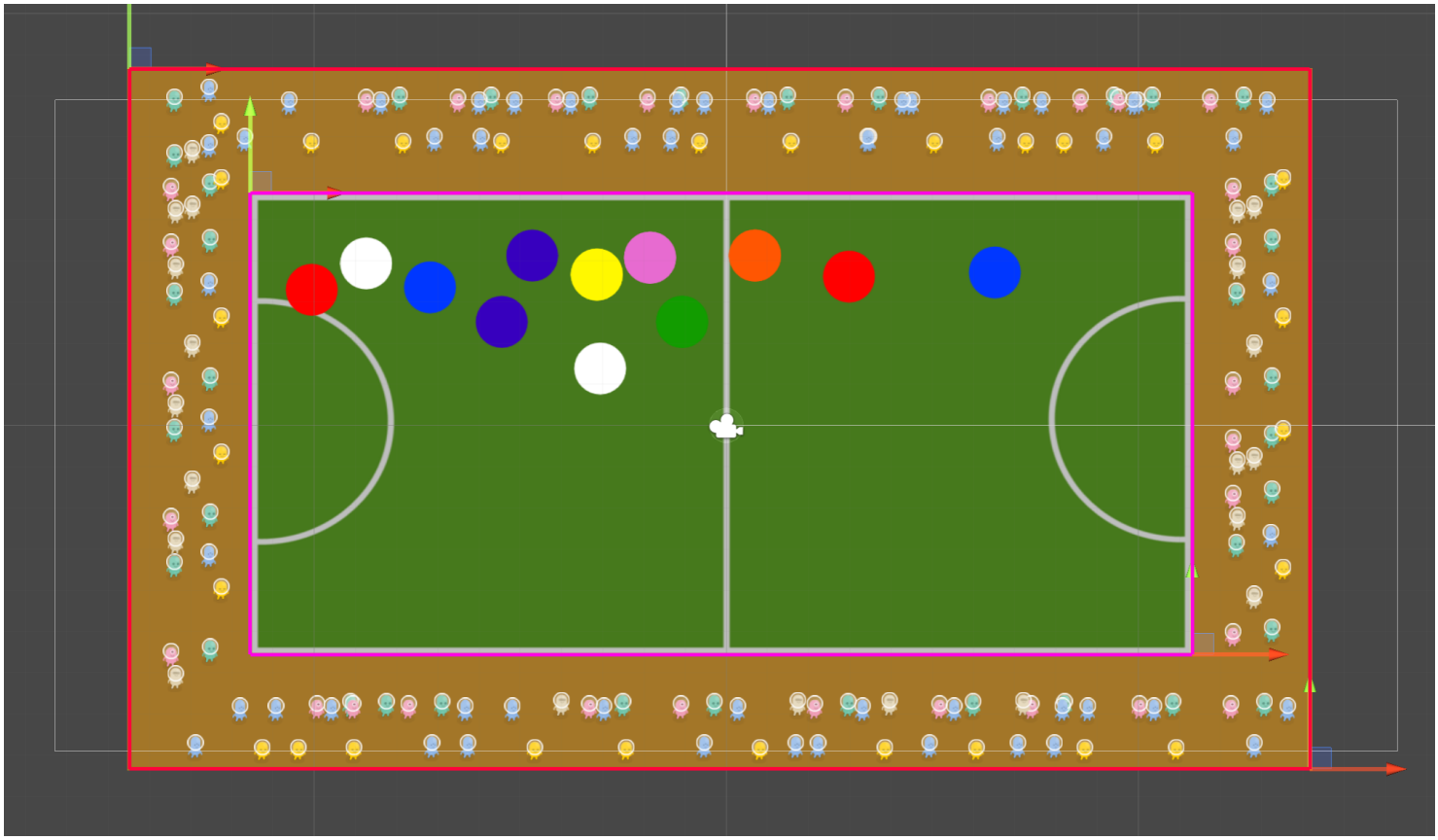
Resolution Magic 2D repositions and resizes the orthographic (2D) camera to fit to the area that you have told it needs to be visible, and does so every time the screen's resolution changes (e.g. when a window is resized or when switching between portrait and landscape modes).

Just design your game and let the magic happen!
Two key items in the ResolutionMagic prefab do the magic:

> *AlwaysDisplayedArea*
> *The part of your game area that is always displayed regardless of the screen. In games with a moving camera, this defines the camera size.*
> *MaximumBounds*
> *The entire area that includes game content, including extra content that may display depending on the screen ratio. Resolution Magic ensures that your player doesn't see anything outside this area.*

### An Example

Think of your game's area as three levels like in the image below. The area inside the pink rectangle will display in full on every device. The area between the pink box and the red box may display in part or in full depending on the screen shape. Anything outside the red rectangle will not display at all.



> *Place anything your player MUST see within the AlwaysDisplayedArea, and use the MaximumBounds for 'nice to have' or extra content some players may see depending on their screen shape.*

Note: You can force all screens to display exactly the same content by setting the background and canvas to be the same size and shape (and have either black bars or incidental content, including UI buttons, outside the canvas area).

**OK…sounds great so far…but do I need to be a Unity guru to use it?**

Resolution Magic 2D is plug-and-play. You just need to define the rectangle sizes for the AlwaysDisplayedArea and MaximumBounds. There are detailed instructions in this documentation for the more advanced features, and several sample scenes to get you started.

This document covers everything, and the scripts are commented in detail so you can understand how the plugin works, and tailor it to your needs.

# Migrating from Earlier Versions

If you have a project using Resolution Magic 3.x (or earlier), you may need to make changes to your project if you upgrade to Resolution Magic 2D 4.0.

> *Read the following section before deciding if you should upgrade to v4.0 for projects already using Resolution Magic 2D 3.x or earlier. If your project is large it may require some work to upgrade.*

## The Main Prefab

The main prefab has been changed to move the boundary items into the same gameobject as the Resolution Manager script. You will need to re-configure the Always Visible and Max Bounds rectangles in your scenes with the new prefab.

> *Also note that certain settings have been removed from the Resolution Manager script as they are no longer needed for the new camera resizing technique in Resolution Magic 2D 4.0.*

## Code Changes

> *If your project does not include code to directly interact with the Resolution Manager script (or other scripts in the asset) you will not need to make any changes to your code.*

The main change in the code is that references to 'canvas' have been changed to 'alwaysvisible' throughout the codebase. This is for clarification and to avoid confusion with Unity's UI canvas.
Some examples of the changes:

**Version 3.x**
CanvasHeight
ZoomToCanvas()
CanvasLeftEdge

**Version 4.x**
AlwaysVisibleHeight
ZoomToAlwaysVisible
AkwaysVisibleLeftEdge

Another change that may impact your project is that the camera changing functionality is now instantaneous instead of taking place over several frames. If you have implemented any code that waits for a resolution refresh you can remove that code.

**New Functionality**
There is some new functionality in v4.0 to make some scenarios simpler:

You can now set the camera to use one or more transforms in place of the Always Visible Area. There are two options for this in the Zoom To option of the Resolution Manager script:

- Focus on Transforms
- Focus on Tilemaps

Each option requires one or more transforms to be selected in the Transforms To Focus On option. Depending on which Zoom To option you select, the asset will calculate the bounds of all the renderers or all the tilemaps in the selected transforms (including children of the selected transforms). See the code reference for more detais.

# Quick Start

This will get you up and running quickly, but we recommend at least playing around with the sample scenes first to get a feel for how Resolution Magic 2D works.

## Quick Start

Import

1. the Resolution Manager asset into your project.
2. For each scene you want to manage the resolution in:
    a. Add the ResolutionMagic prefab
    b. (OPTIONAL) Adjust the ResolutionManager script settings in the Inspector
    c. With the ResolutionMagic gameobject seected, adjust the Always Displayed and Max Bounds to suit your game's content.

       NOTE: you must have Gizmos enabled in the Scene view to adjust the area

You can skip to the code section at the end of this document to see code examples.

## Test it

The quickest way to test the functionality is to run your scene in the Unity editor and detach the game screen, select *Free Aspect* and resize the screen to different shapes and sizes. There are various settings in the Inspector window that let you customise how the resolution is changed. Experiment to get it how you like.

# Sample Scenes

## Sports

This sample scene shows the main scenario of keeping a part of the game area visible at all times. To see the screen refreshed in real time, simply set the Unity Game window to 'Free Aspect' and change the size. The sports court will stay fully visible at all times.

## Things to Try

- Make sure the Game window is set to Free Aspect to see how the screen changes as the resolution/ratio changes.
- Try setting the option to focus on transforms and choose a few of the balls to focus on. The camera will always keep those balls in view.

## Black Bars

This sample scene shows how black bars can be used to force the exact same screen ratio on all devices. Black bars are automatically shown whenever the screen shape is different from the Always Visible area. Note that some of the game content can be hidden by black bars, which is useful for competitive games where you don't want players with wider/taller screens to be able to see things other players can't.

## Focus on Objects

This scene demonstrates the scenario of the camera keeping objects on screen at all times. Move the character with the arrow keys and watch as the camera adjusts to keep the character and tree in view at all times.
Note:

- The objects must stay within the Max Bounds area for the camera to keep them in view.
- This functionality will now work if another script (such as a camera follow script) overrides the camera position.

## Side Scrolling

This simple demo uses a camera follow script that keeps the player centred, but will not move outside of the Max Bounds area.

## Space

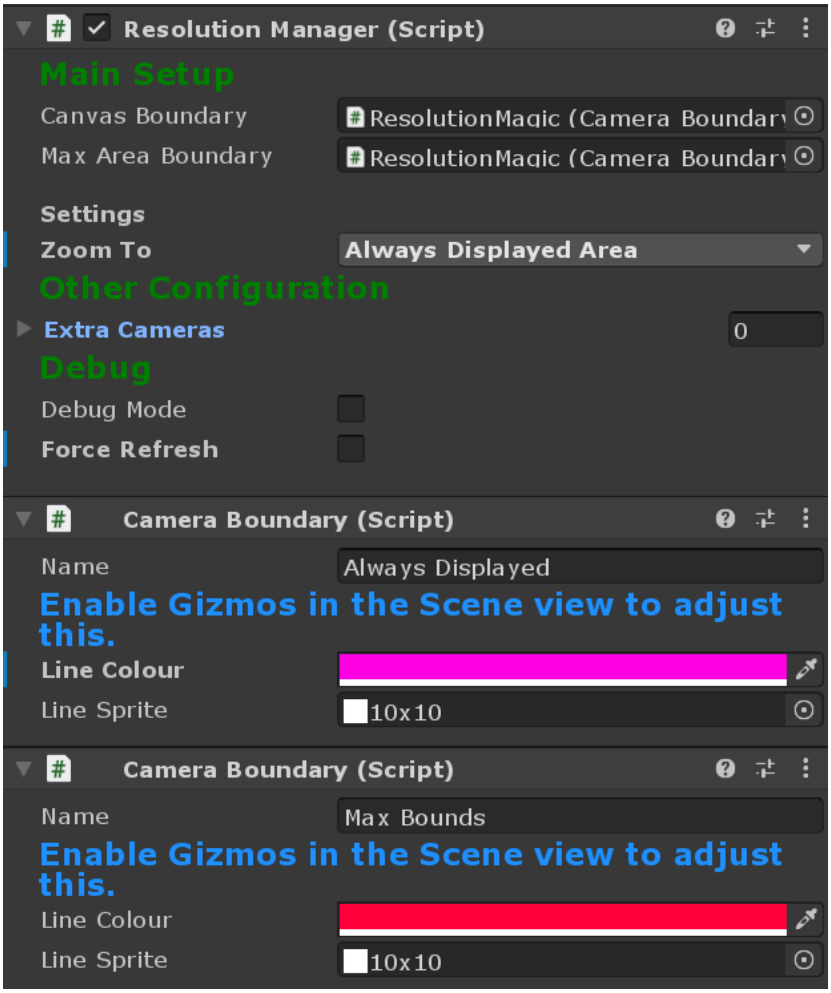
This scene demonstrates the Always Visible and Max Bounds areas. Pan the camera with the arrow keys and adjust the Game window to get an idea of how it works. You can click the button to switch between the two types of zoom.

# Overview

> *This overview gives a top-level description of how the pieces fit together. There is no code in this section.*

## The ResolutionMagic Prefab

The prefab contains an instance of the Resolution Manager script and two Camera Boundary instances, which are used to set the sizes of the Always Displayed area and the Max Bounds:




## The Resolution Manager script

This script is the heart of Resolution Magic. It manages everything. Most of the magic is done automatically according to the settings you choose (see the next section for details).
With the default settings, ResolutionManager will automatically update the camera to fit y0ur content as per your settings any time the screen resolution or ratio changes.

## Public Properties and Methods

There are several public properties and methods you can access on the ResolutionManager script. These are detailed later in the documentation.

## The Always Displayed Area

By default the area defined by this rectangle will be fully shown on all screens, and will be shown as large as possible without any of the content going off the screen edges. In other words, the camera will be zoomed in as close as possible to this object without zooming so far that any part of the object goes outside the screen area.
In games where the camera moves, this will set the basic camera view.

### The Max Bounds Area

This area is content that will show on screens that are not the same shape as your Always Displayed area. For example, if your Always Displayed area is wide, but the player views it in portrait mode, you cam place extra content in the Max Bounds area above and below the Always Displayed area that the player will see.
Another use for the Max Bounds is to define the absolute limit of the camera with a camera that moves, i.e. the camera will not be allowed to move far enough to show anything outside the Max Bounds area.

> *Note: The Max Bounds area must be larger than the Always Displayed area.*

### Hard Border

This helper prefab adds a physical collider around the edge of the screen or the Always Displayed area, which will prevent objects from moving past it. You can see an example of this in the Sports sample scene, where a hard border is used to keep the balls within the sports court area.

### Black Bars

You can add optional black bars (explained in more detail later in the documentation) for certain use cases. For example, if you want to show some content in a specific aspect ratio (like a cutscene), or if you want to make sure that all players see exactly the same game area regardless of their screen size in a game where players with a wider/taller screen might otherwise have an advantage.

# The Resolution Magic Prefab

You must place an instance of this prefab in every scene in which you want Resolution Magic 2D to do its magic. The prefab contains a single gameobject with an instance of the Resolution Manager script and two Camera Boundary instances, one for the Always Visible bounds and one for the Max Area boundary.

## The Resolution Manager Script

The Resolution Manager script holds the settings and options for the camera sizing you want in your scene, and also includes various public methods and properties available to call from your own code for more complex and nuanced functionality.

## User Settings

> *Some settings are only available when specific Zoom To methods are selected, so not all settings will appear all the time.*


## Bounds Setup

These are references to the two instances of Camera Boundary and should not be changed ordinarily. Enable Gizmos in the Scene view to see and edit the boxes that form these areas to ensure your game camera will see exactly what you want.

## Camera Sizing Method

## Zoom To

This is the most important setting as it determines the camera behaviour for the scene. The most common option to choose here is Always Displayed Behaviour, which will make the camera maximise the view of the Always Visible area and make sure it is visible in full, and as large as possible, on all screen shapes and sizes. The options are:

- Always Visible Area - show the always visible area in full and as large as possible on all screens
- Maximum Bounds - show as much of the maximum bounds area as possible, and avoiding showing any area outside the maximum bounds if possible. The Always Visible area will also be fully visible.
- Focus on Tilemaps - treats one or more tilemaps as an Always Visible area, overriding the Always Visible boundary.
- Focus on Transforms - treats one or more transforms (and all the renderers within) as an Always Visible area, overriding the Always Visible boundary.
    - Note: this option replaces the similar 'Keep Objects on Screen' option from earlier versions.

### Zoom Style

- Instant - the camera is adjusted immediately when a resolution change is detected (within a single frame)
- Smooth - the camera will be moved and zoomed in a smooth transition. The speed of this is determined by the Transition Speed setting.

### Other Options

- Centre on Always Visible - (only available when Maximum Bounds is selected) centres the camera on the Always Visible area instead of on the Max Bounds area.
- Transforms to Focus On - (only available when Focus on Tilemaps or Focus on Transforms selected) the transforms that hold the objects to focus the camera on. This is a list of transforms, and the asset will calculate what it needs from the tilemaps or renderers within.

## Additional Settings

### Extra Cameras

Here you can add more cameras to inherit the positioning and zoom settings from the main camera. Whenever the camera is adjusted by the Resolution Manager, these cameras will be set to the same values.

### Auto Refresh

When enabled, the camera will automatically and instantly adjust to any change in screen resolution/ratio. Typically, this should be left enabled, but you may want to disable it in some scenarios. This setting is publicly available to scripts.

### CameraZoomChanged Event

Whenever the Resolution Manager script changes the camera, it raises the CameraZoomChanged event, which sends two parameters (previous screen size and new screen size) to any listeners to notify

of the screen change.
To subscribe to this event, add the following code to your script(s). Subscribe to the event with the following line of code, which you would usually put in your Start() method:

```
ResolutionManager.Instance.CameraZoomChanged += CameraZoomUpdated;
```

And add this method to respond to the event:

```
void CameraZoomUpdated(float oldSize, float newSize)
{
    // code to react to the event goes here, e.g.
    Debug.Log($"Camera size changed from {oldSize} to {newSize}");
}
```

### Automatic Camera adjustment

Camera adjustment is automatic. There is no code or intervention needed by you to trigger this.
The camera is adjusted at the start of the scene, and also any time the screen resolution changes (if that option is enabled), such as when a Windows Store app has its screen area changed or when a player changes their tablet or phone from landscape to portrait.

### Public methods

The following public methods can be accessed from your code to trigger certain actions.

### Move the Camera

Resolution Manager can move the camera if you request it from your code. There are two movement methods available, and each movement can be *safe* or *unsafe*. When Resolution Manager moves the camera with safe movement, it means that the camera will only move to the new position if it doesn't reveal any area outside of the MaximumBounds. Unsafe movement will simply move the camera without checking.

The two methods to move the camera can move in a specific direction or to a new position:

```
public void MoveCameraInDirection(CameraDirections direction, float moveDistance, bool
moveSafely=true);
public void MoveCameraPosition(Vector2 newPosition, bool moveSafely=true);
```

By default both methods will move the camera safely.

You can easily integrate an existing camera move/camera follow script by replacing the code that moves the camera with a call to one of the above methods. That way you can keep your existing camera movement logic, but also ensure your camera stays within your game's bounds.

### Refresh the resolution

Although the resolution is refreshed automatically by default, you can force a change at any time (for example if you have disabled automatic refreshing). You might use this to have different camera settings depending on events in your game or for specific screens (such as using canvas zoom for phone screens and screen zoom for tablet screens).

Example:

```
ResolutionManager.Instance.RefreshResolution();
```

### Properties

Resolution Manager contains several public, read-only properties that you can access from your game code if needed.

### CalculatedCameraSize

This contains the orthographic camera size that Resolution Magic calculated to scale the camera the last time the resolution was refreshed. This is the same value you would get from:

```
Camera.main.orthographicSize
```

The difference is that this property stores the last calculated value from Resolution Manager, so you can treat this like your game's default camera scale for the current screen it is being played on. For example you may want to zoom the camera in to highlight something in your game, and then set the camera size back to what Resolution Magic previously calculated (if you do need to zoom the camera in or out in your game you should disable the automatic resolution check as it might interfere and reset the resolution when you don't want it to.

You could also use this value as a camera size for your whole game if you don't need to have the Resolution Magic prefab in every scene. For example, in your game's loading scene or first game screen you can let Resolution Magic choose the scale based on a certain scene, then store this value somewhere to use for setting the camera size in all scenes. You will lose some functionality using it this way, but it doe allow you to get things working very simply for certain games.

### Screen edge properties

These properties return the furthest point on the specified screen edge as a float. For example, ScreenEdgeLeft returns the left edge of the screen's x-axis value.

```
// returns the point at the middle of the left edge of the screen.
Vector2 leftCentre = new Vector2(ResolutionManager.Instance.ScreenEdgeLeft, 0)
```

- ScreenLeftEdge

- ScreenRightEdge
- ScreenTopEdge
- ScreenBottomEdge

You can combine two of these properties to get a corner point:

```
// top left corner
Vector2 topLeftCorner = new Vector2(ResolutionManager.Instance.ScreenEdgeLeft,
ResolutionManager.Instance.ScreenEdgeTop);
```

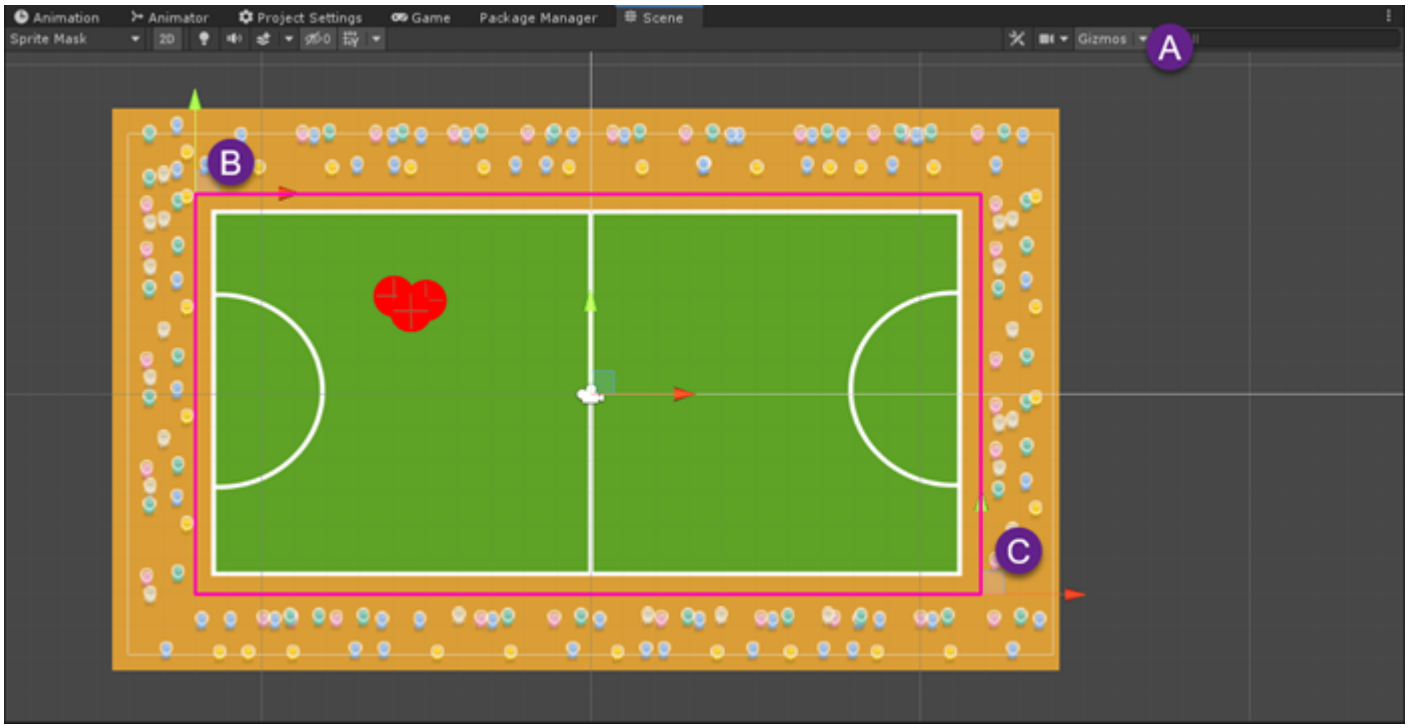
### Always Visible edge properties

These properties return the furthest point on the specified canvas edge. This may or may not be equal to the corresponding screen edge depending on the screen ratio.
Note: add the camera position to these values if you want the actual position on screen.

- AlwaysVisibleLeftEdge
- AlwaysVisibleRightEdge
- AlwaysVisibleTopEdge
- AlwaysVisibleBottomEdge

## The Always Displayed Area

Use the Always Displayed Area to define your core game screen area. Anything within its bounds is guaranteed to be displayed on any screen your game is played on.
To adjust this size, select it in the scene Hierarchy, then drag the top left and bottom right corners to resize and shape the rectangle.
a. You need to turn on Gizmos toenable reshaping the area
b. There is one handle for the topleft corner
c. And one handle for the bottomright.




> *Some content outside the designated area may display depending on the screen ratio and your settings. For example, if your area is wide, but the game is played on a screen that is less wide, some content outside the Always DIsplayed area will show above and below to fill the screen.*

### AlwaysDisplayedArea Quick Notes

- All content within the Always Displayed Area will be visible to all players
- Resolution Magic will centre the screen on the Always Displayed Area when your scene starts. The camera size will stay at the set size unless you decide to change the camera size later.
- You can resize the canvas to any size/ratio you want for your game by adjusting its scale in the inspector (and you can use different sizes and shapes in difference scenes). You can drag a new canvas shape, but be sure to reset its position to (0,0) to ensure you are designing your layout correctly

### The MaximumBounds

The MaximumBounds defines the full content in your scene. Resolution Magic will never show any content outside of this area regardless of screen size/shape as long as you provide enough content to cover any screen when the AlwaysDisplayedArea is fully zoomed (unless you override the default behaviour, e.g. by moving the camera unsafely). This area should contain content that the player doesn't NEED to see (as that should be in the AlwaysDisplayedArea) but that the player *may* see depending on their screen.

Typically MaximumBounds would be quite a bit larger than your AlwaysDisplayedArea so that there is content to display on the edges when your canvas is a different shape/ratio from the player's screen. For games where the camera moves, you can use the MaxiummBounds to define the entire level area (e.g. make it very wide to cover the entire content of a side-scrolling level) or you can merely use it to aid in camera zooming. See the Side Scrolling demo scene for a demonstration of this.

> *Make sure your maximum bounds area is large enough to accommodate all screen shapes to avoid any blank areas displaying outside (unless that is what you want). Test by making the Game window in Unity very wide and very tall to imitate the most extreme shapes of screen.*

## Hard Border

This is an optional feature. When the prefab is added to a scene, a box is created around the edge of the screen or canvas (you can choose). This box is solid, in that it has a collider forming a wall on the screen edge (each edge can be aligned and sized to either the Always Displayed Area or the screen edges) that physics objects will collide with.

You can set each edge to be on or off individually, and customize their colliders to your needs.

# Black Bars

A major use case for Resolution Magic 2D is to avoid 'black bars' when your game is played on a screen with a different ratio than you designed for. But there are some use cases where you might prefer to have black bars, such as:

- You want to show a cutscene in a specific aspect ratio like a movie
- You want to be make sure all players see exactly the same content so players with different screens don't get an advantage.

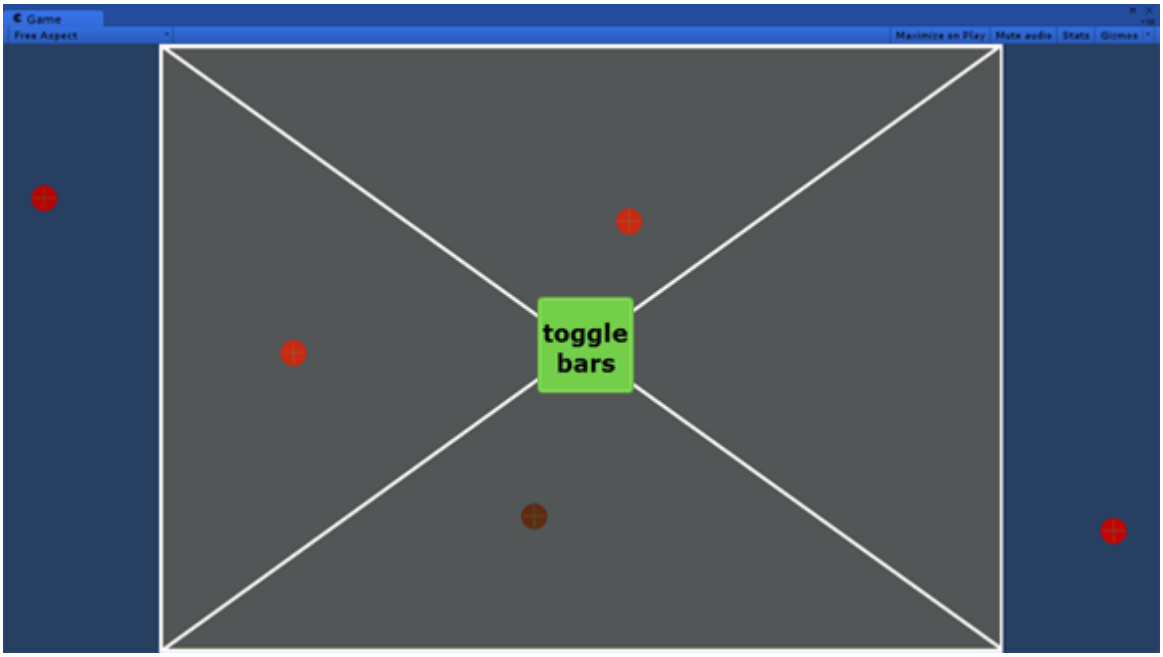
The Black Bars prefab can be added to the camera to fill any empty space in the screen with ‘black bars’ (like when you watch an old 4:3 TV show on a widescreen TV).

This should only be used if your players absolutely have to all see the same screen region (i.e. the Always Displayed Area) and you don’t implement any way to prevent extra content showing in extra screen space available on wider/taller devices.

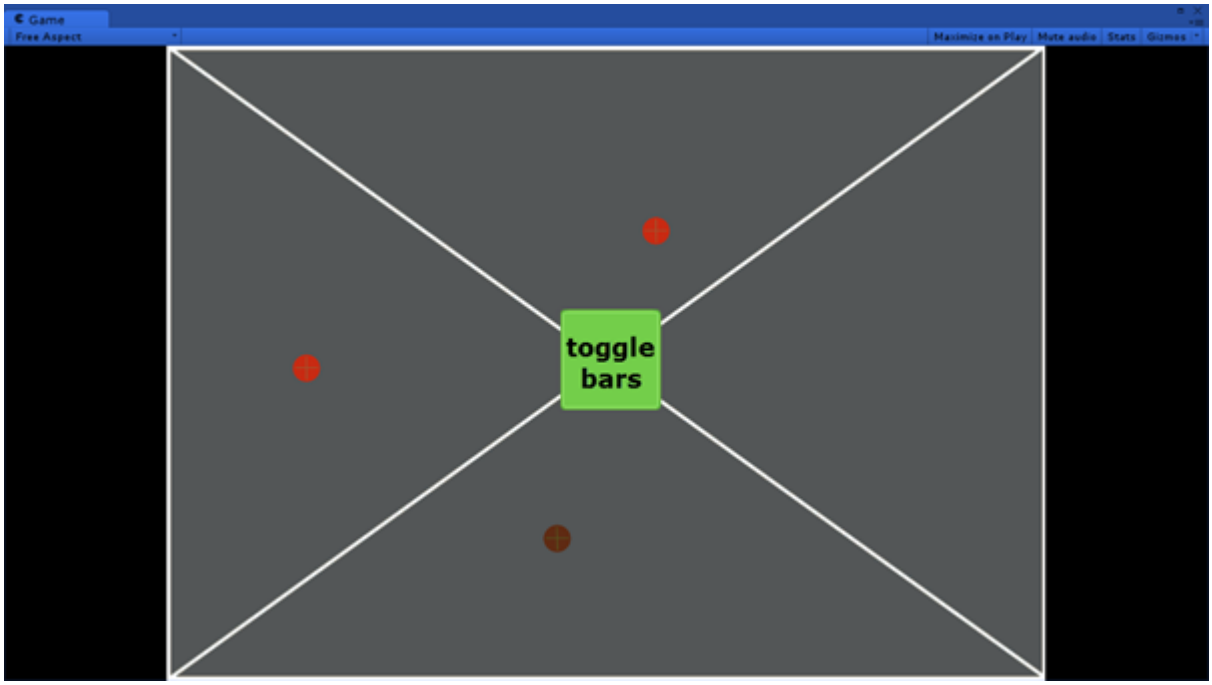
To use, just place the Black Bars prefab in your scene *as a child of the camera*.

## Example

In this example, the grey area represents the Always Visible content that all players will see, and the blue area is part of the Maximum Bounds that players will only see if their screen is wider than the Always Visible area.




Note that some players will see the extra red balls to the sides. In a competitive game you might not want this behaviour, so you can hide that extra content behind black bars to get the following result:




Now all players see exactly the same content and players with a wider screen don't have an advantage.

> *Many players hate black bars, and eliminating them a major function of Resolution Magic 2D. Only use black bars if you absolutely must prevent some players seeing more content than others, such as in multiplayer games when it might give an unfair advantage.*

# Typical Use Cases

This section contains some example scenarios and how you would implement them with Resolution Magic 2D.

## Typical Use – Adapt the Camera in Each Scene

The typical use case for Resolution Magic 2D is to set the correct camera size automatically for every scene, and to adapt if there is a change in screen resolution/ratio.

- Put a copy of the ResolutionMagic prefab in each scene and setup according to that scene's needs
    - If all your scenes require the same settings you can just create a copy of the prefab and change its settings to your game's defaults and use that.
    - Some games may need different camera sizes per scene (e.g. a game that needs to show the whole level on screen at once, but may have differently shaped levels). For these you should set the AlwaysDisplayedArea separately for each scene.

To ensure the resolution refreshes when there is a change in the device, leave Auto Refresh enabled on the prefab.

## Single-Screen Games

For games that have a single-screen type setup, such as individual levels that don't scroll, you can setup the Resolution Magic 2D asset to zoom in three ways depending on what suits your game best:

- Focus on a tilemap (or multiple tilemaps)
- Focus on one or more transforms and their renderers
- Use the standard Always Displayed area.

When using tilemaps and/or transforms, simply select the appropriate option from the Zoom To field and drag-and-drop the transforms into the list. The combined bounds of all the renderers or tilemaps in those transforms will become the basis for the camera sizing.

## Every Player Should See Exactly the Same Content – Never More (or less)

In some games you may want to prevent players from seeing more content than other players when they have a differently shaped screen.

For example a war game may give a player with a taller screen an advantage because they will be able to see a bit more of the map above and below the AlwaysDisplayedArea.

To get around this, you can enable black bars to block out any content that is outside the Always Displayed Area. Simply add the Black Bars prefab to your scene as a child of the camera. The included script will automatically fit the black bars to the edges of the Always Displayed area.

## Keeping Objects on the Screen

This scenario is for when you want your camera to always show - and focus on - one or more objects. The camera will move and resize to always fit the selected object(s).
To implement this, simply choose the Focus on Transforms option and also select the Update Every Frame option.

### Constant Camera Size

If your game should have a constant camera size throughout, you can use Resolution Magic 2D to determine the correct camera size for the current screen, then save that value in a variable and use it to set the camera size at the start of every scene.

Note: with this technique your camera will not resize automatically to adapt to a change in screen resolution, such as a phone being rotated from landscape to portrait or a window on a PC being resized.

- Put the Resolution Magic prefab in the first scene (or possibly in a menu screen that displays before the game starts). Set the desired camera view by adjusting the Always Displayed Area.
- You can force a refresh of the camera size by calling

```
ResolutionManager.Instance.RefreshResolution();
```

- Once the resolution has been set to match your Always Displayed Area, you can get the current camera size and save it somewhere (you will need to store it in a way that is accessible across scenes):

```
    var cameraSizeForCurrentDevice = ResolutionManager.Instance.CalculatedCameraSize;
```

- Now for any new scene that needs the camera to be the same size, you just set the camera to the stored size:

```
    Camera.main.orthographicSize = cameraSizeForCurrentDevice;
```

## Code Reference

### ResolutionManager.cs

#### Move the Camera

You can use public methods on the script to move the camera while letting the Resolution Manager keep your camera from moving outside the desired view.

```
public void MoveCameraInDirection(CameraDirections direction, float moveDistance, bool m
oveSafely = true)
```

CameraDirections is an enum with up, down, left, and right possible values. moveDistance tells the camera how far to move in the direction given. moveSafely is optional (this will default to true as of v3.0, but previously defaulted to false), and when true the camera will only move if doing so does not reveal area outside of your game's MaximumBounds area.

```
// move the camera 0.5f to the left, but only if doing so doesn't reveal area outside my
game design
ResolutionManager.Instance.MoveCameraInDirection(CameraDirections.left, 0.5f, true);
```

You can also directly set the camera's position with MoveCameraPosition.

```
public void MoveCameraPosition(Vector2 newPosition, bool moveSafely = true)
```

#### Refresh the Camera Resolution

> *This is rarely needed, but has been left in for compatibility and for edge cases.*

```
public void RefreshResolution()
```

Simply call the above method to force a resolution refresh. This is redundant if Auto Refresh is enabled, as the resolution is refreshed automatically whenever it changes if that is the case. You may have use cases where you want to refresh manually, and you can use this method for that.

```
// reset the resolution
ResolutionManager.Instance.RefreshResolution();
```

#### Show and Hide Black Bars

```
public void TurnOnBlackBars()
public void TurnOffBlackBars()
```

You can call these methods to show/hide black bars. They will only have an effect if the Black Bars prefab has been added to a camera in the scene.
Note: black bars will enable automatically by default. You only need to call the above methods if you want to toggle bars on and off, which should be unnecessary. When black bars are enabled the camera will centre on the canvas, so take that into account if you use these methods.

### BlackBars.cs

#### Enable and Disable Black Bars

The BlackBars script has a single public property that can be used to toggle the bars off and on:

```
public bool Enabled
```

You must have a reference to an instance of the script to change this. Once you have a reference, you simply change the value:

```
blackBarsScriptInstance.Enabled = true;
```

#### Code Notes

There are some public methods you can call manually that are not intended to be used that way. These are undocumented, but are explained via code comments.

There are some extra generic scripts in the example scenes used for simple input, movement, and so on required in the demo scenes. These scripts are not considered part of the asset, and are not supported nor intended to be used in a real Unity project.

## Troubleshooting

Resolution Magic 2D is mostly plug-and-play, so hopefully everything works fine for you. Here are some things to check if you have trouble getting things to work.

- Version 4.0 is a major version update, and many things work differently. It is not recommended to update to this version if you already have Resolution Magic deeply integrated into your game. Please read the documentation and changelog closely before deciding to upgrade an existing project.
- (Version 2.0 and older) Any error that mentions the script EditorResMagicEditor.cs can be fixed by deleting the EditorResMagicEditor.cs script. This script is from an older version of the asset, and is no longer compatible with Unity. If upgrading to the latest Resolution Magic doesn't remove the script automatically, you will need to delete it yourself.
- Make sure you have the ResolutionMagic prefab in your scene. Nothing will work without it.
- Re-import the asset into your game to reset everything.
- Make sure the scripts are attached where they need to be. ResolutionManager is required in all scenes where you want to manage the resolution (as part of the prefab).
- Double-check your settings, and check this documentation to make sure you're using the settings correctly.
- Resolution Manager works as a singleton class, which means you don't have to instantiate it at all. You simply access it in your code by referencing it in the following way:

```
ResolutionManager.Instance.[method or property name]
```

**And of course, don't hesitate to contact support@grogansoft.com for any help, questions, etc.**

# Change Log

## Version 4.0 - May 2023

### Major Update

> *If upgrading a project already using an older version of Resolution Magic 2D, please uninstall the old version before installing version 4.0.*

- Camera sizing is now more precise
- Camera (re)sizing now happens within a single frame.

The way that Resolution Magic 2D changes the camera to fit your game content has been completely rewritten. Due to the improvements in the way the camera is changed, some settings options have been removed as redundant.

- Settings for auto refresh are removed. You can still disable auto refresh, but when it is enabled (which is usually should be), the screen will automatically refresh instantly whenever a resolution or ratio change is detected.
- The settings for Zooming are removed, as zooming is no longer required for sizing the camera.
- Some improvements to the main Resolution Magic prefab structure.
- Some minor changes to the sample scenes.

> *Throughout the code, the term 'Canvas' was previously used to describe the 'Always Visible Area'. This has now been changed to avoid confusion with Unity's Canvas and to make its use more clear. Any code you have written to integrate with Resolution Magic 2D may need to be changed to match the new naming.*

### Version 3.1

### Minor Update

Fixed some issues with the black

- bars feature

Bar
sizes are now slightly smaller, as previously the margin of error on
calculating the size would sometimes make a few pixels of black bar show where

a. it shouldn’t

Black

a. bar positioning now works correctly with the new Resolution Magic prefab

Documentation corrections and minor

- update

New logo and links on the Asset

- Store page.

**Version 3.0**

**Major update with BREAKING CHANGES**
It is not recommended to upgrade to version 3.0 if you are deep into a project. Version 3.0 makes fundamental changes to all aspects of Resolution Magic 2D and upgrading will require you to reconfigure any game scenes that use Resolution Magic 2D.
This update includes several ‘breaking changes’ – changes that cause errors and/or stop your game from working correctly if you update over an existing version of this asset.
Note: names have been changed for the Canvas (now called “AlwaysDisplayedArea”) and the Background (now called “MaximumBounds”). These terms are more intuitive and don’t overlap with Unity-specific terms (e.g. the UI Canvas).

**New Core Prefab (breaking change)**
The main Resolution Magic prefab has been updated and improved. Note: this completely replaces the old prefab. The main differences are:

- Some new customisations

Easy bounds drawing with custom box

- colours to set your viewable canvas and max game area

Choice between smooth or instant

- resolution changes

**Removed: UI Functionality (breaking change)**
Unity’s built-in UI functionality made Resolution Magic’s UI functionality obsolete long ago, and it has now been completely removed as of Version 3.0.

**New Feature: CameraZoomChanged event**
When the camera zoom is changed by the Resolution Manager script, an event is now raised, which you can subscribe to from your own code. When the event is raised, it will provide the previous camera size and the new camera zoom size.

**Black Bars Changed (breaking change)**
The black bars functionality has been removed from the main ResolutionManager script and now works separately. Simply add the black bars prefab to a scene to use it as before.

**New Demo Scene**
A new side-scrolling demo scene has been added to better illustrate how to use the asset in a game with a moving camera.

**New Property: CalculatedCameraSize**

Returns the last camera size that the Resolution Manager script calculated. This can be used to store the ideal camera size for the current scene or as a way to return to the correct camera size after zooming the camera in or out during gameplay.

**Refactored Code (breaking change)**
This version of the asset has a lot of refactored code, with some functions and variables renamed for clarity and consistency. Some aspects of the way the asset changes the resolution are also changed. If you have written code to extend or directly interact with Resolution Magic 2D, this refactoring will potentially require you to update your code. If in doubt, it is recommended to keep the existing version of the asset on your current projects.

**Version 2.1**

**New inspector field: Leave Canvas Sprite Visible at Runtime**
A new inspector field called ‘canvasSpriteVisibleAtRuntime’ has been added to the *ResolutionManager* script.
When this is enabled, the sprite attached to the Canvas object (if there is one) will remain active and visible at runtime, otherwise the sprite it automatically disabled (i.e. hidden) at runtime.
The default canvas sprite contains a cross pattern and shading to make it easy to see the canvas shape when building your scene, but in most cases you would not want to leave this sprite visible while playing the game.
The default value is FALSE.

**Removed: Editor Script**
The script EditorResMagicEditor.cs has been removed from the asset. This script caused a conflict with changes to Unity. The script did not affect functionality other than a refresh button in the inspector.
If you have errors mentioning that script when compiling your project, you can safely delete the script file if it is not automatically removed by upgrading to Version 2.1.

**Demo scene change: Sport**
The sports demo scene has been changed slightly to better demonstrate the difference between the zoom types. Please refer to the instructions in the Sports folder for full details.

**Version 1.2**

**New feature: black bars**
You can now add ‘black bars’ along the screen edges if you want to force all players to see exactly the same content and don’t use your own background to fill extra space.
Simply add the Black Bars prefab to the camera, and they will be enabled by default. You can turn the bars on and off via code.

**New sample scene**
A sample scene has been added to show the new Black Bars feature.

**Version 1.11**

[No changes]

Added a Unity 5 compatible version with minor code changes for Unity 5 compatibility. No feature changes.

**Version 1.1**

**New feature: multiple cameras**

You can now set multiple cameras to be zoomed by the ResolutionManager script. This is useful if, for example, you use a separate camera to show UI and need it to match the main game camera.

**Bug fix: AlignedObjects don't respect settings for show/hide**

There was a bug preventing manually showing/hiding AlignedObject items (i.e. not from the ResolutionManager script) using the ShowThis() and HideThis() methods. New methods have been added for this functionality (see below).

**New feature: improved options for showing/hiding UI**

New methods introduced to force the UI objects (with the AlignedObject script attached) to show or hide regardless of settings. These methods allow items to be hidden/shown without the transition animation (i.e. immediately) and also to show/hide regardless of any settings:

ShowNow() / HideNow() – will show/hide the UI element regardless of settings to ignore the manager

ShowInstant() / HideInstant() – immediately move the item to the hidden or displayed position with no animation, regardless of settings to ignore the manager

**New feature: hide UI at start**

A new checkbox is available on AlignedObjects, called StarfOffScreen. When active, the AlignedObject will be off the screen (in its hide position) when the level starts.

Note: you must still place the item within the canvas area when designing the layout.

**Notes**

Some changes have been made to properties in the ResolutionManager script; they must all be accessed via ResolutionManager.Instance now.

If any of your code is broken by the upgrade you will simply need to change any property references to reflect the changes.