Version 3.1

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)

And:

- NO MORE black bars (unless you want them – they are included as an optional feature)
- NO MORE stretching
- NO MORE fiddling around to test on different resolutions.

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

Just design your game and let the magic happen!

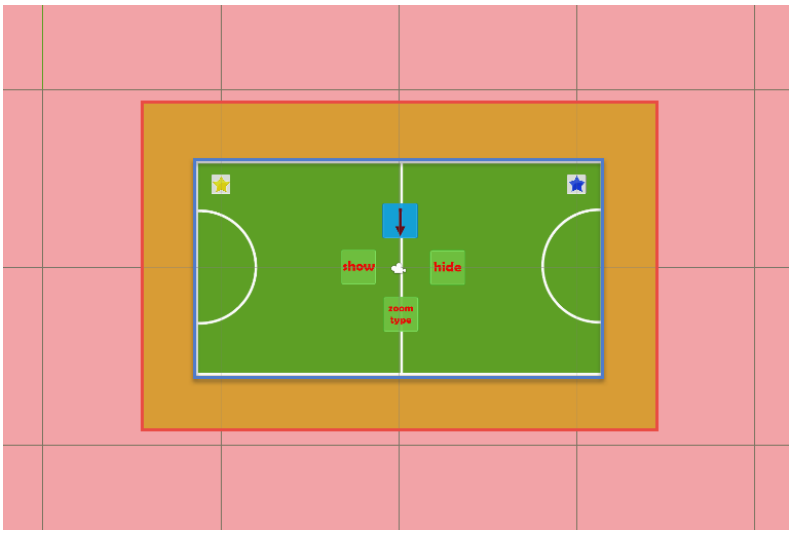
Two key items in the ResolutionMagic prefab do the magic:

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.



THE DIFFERENT AREAS

In Figure 1 there are three distinct areas of a sports game screen. The green area is set to display in full on every device. The orange area (background), may display in part or in full depending on the screen shape. Lastly, the pink area is anything outside the background that won't be displayed.

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). As of version 1.2, Resolution Magic lets you implement 'black bars' by adding a simple prefab to your game camera.

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.

Getting Started

If you want to jump right in, you can follow the Quick Start Guide. You can get Resolution Manager 2D up and running in a couple of minutes.

You should read the brief Overview section, which explains the basics of how the asset works.

The rest of this guide details the workings of the plugin, including code samples for all the methods and properties.

Because Resolution Magic 2D is intended to be 'plug and play' you don't need to add much code to your game. You personalise the behaviour via the script properties, and call a couple of methods when required. All the source code is included so you can extend it if you want.

Change Log

Version 3.1 (February 20th 2021)

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 it shouldn't
    - Black 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.

Quick Start Guide (Advanced Users)

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

1. Import 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. Adjust the AlwaysDisplayedArea and MaximumBounds to suit your game's content
        i. NOTE: you must have Gizmos enabled in the Scene view to adjust the area sizes.

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

IT'S EASIER TO SET UP RESOLUTION MAGIC FOR A NEW PROJECT, BUT ADDING IT TO AN EXISTING SCENE IS ALSO QUITE QUICK AND EASY

Test it

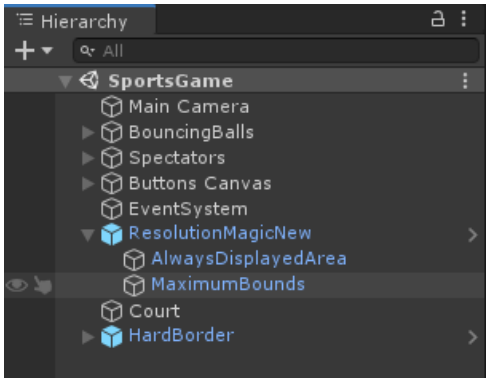
The quickest way to test the functionality is to run your scene in the Unity editor and detach the game screen, select *Free Aspect* and resize the screen to different shapes and sizes. There are various settings in the Inspector window that let you customise how the resolution is changed. Experiment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

Overview

Read this overview to get a top-level understanding of how the pieces fit together. There is no code in this section.

The ResolutionMagic Prefab

The ResolutionMagic prefab goes in your scenes. It contains three items:



THE RESOLUTION MAGIC PREFAB IN THE HIERARCHY

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).

Automatic tasks in ResolutionManager

- Adjusts the game view and UI scale as per your settings
- Re-adjusts the game display if the resolution changes (you can also do this manually by calling a function, but there is usually no need)

Public properties you can access

Multiple properties are exposed for you to access if you want to fine-tune your display or add functionality, such as the Canvas edge points. Details in the next section.

The AlwaysDisplayedArea

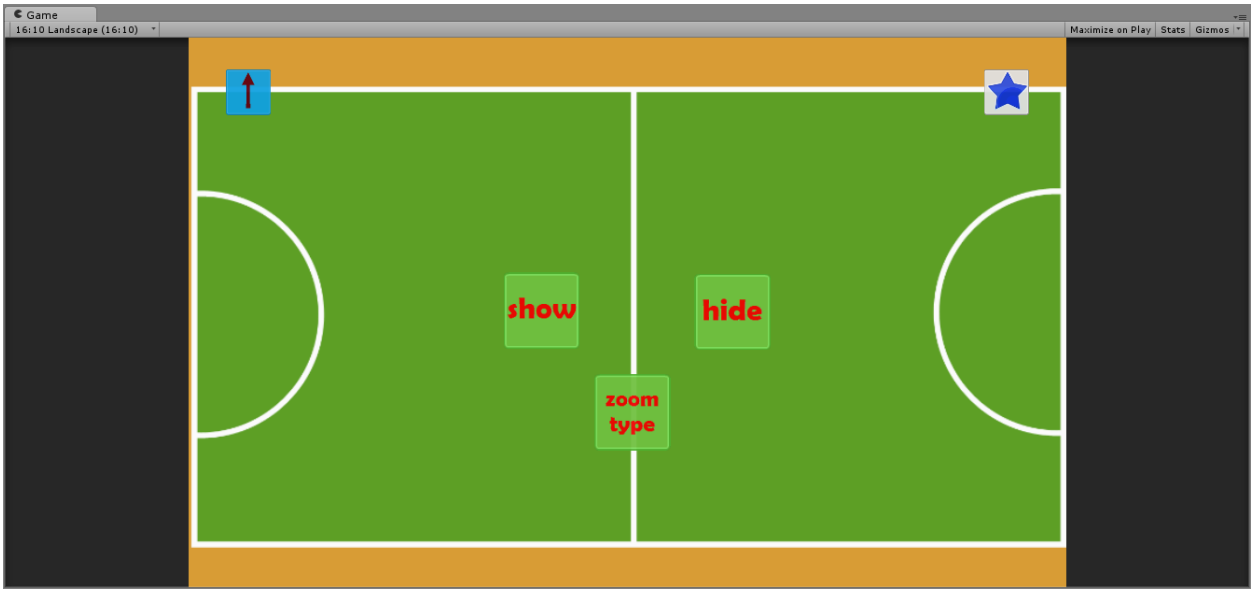
The canvas is a special game object that defines a rectangle of any shape and size representing the content visible to every player.

> ***AlwaysDisplayedArea***
> *All content within the canvas rectangle is visible to all players regardless of their device screen size/shape.*

Think of the AlwaysDisplayedArea as a template for your game screen. This area will be displayed on ALL screens regardless of size or shape, and on screens that are a different shape ratio to the canvas (taller or fatter), any content outside of the AlwaysDisplayedArea will display in addition to the AlwaysDisplayedArea view area to

ensure the screen is filled up and doesn't have 'black bars'. However, the asset does also include an option to use black bars for games that require all players to see exactly the same content.

Area outside the AlwaysDisplayedArea can be extra game content or simply an extension of your background or UI depending on the game type.

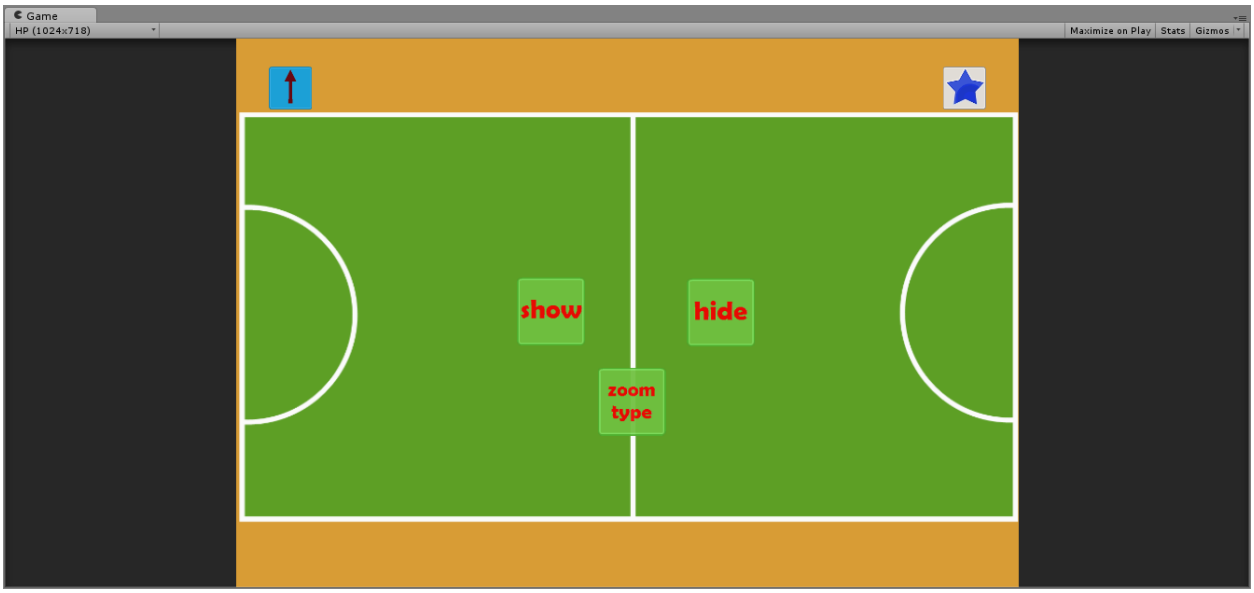


THE GREEN SPORTS COURT FILLS THE ALWAYSDISPLAYEDAREA SO IT IS ALWAYS VISIBLE, THE ORANGE BACKGROUND IS PARTIALLY VISIBLE BECAUSE THE SCREEN RESOLUTION IS TALLER IN THIS EXAMPLE.

MaximumBounds

The MaximumBounds represents your entire game area, some of which may or may not be visible to the player depending on their screen shape and the behavior you choose in the settings. You should include enough content in your Background to ensure the player's screen is always filled with content when the screen ratio does not match the AlwaysDisplayedArea ratio (unless you want to use black bars). This usually means you should have a 'buffer' around your canvas with 'optional' content.

You can also use the MaximumBounds for games where the camera moves. By setting the MaximumBounds around the entire level, you can then ensure that when you move the camera it doesn't stray too far and show content that is not within the MaximumBounds.



ON A DIFFERENT SCREEN RATIO THE BACKGROUND IS MORE VISIBLE

THE MAXIMUMBOUNDS MUST BE THE SAME SIZE OR BIGGER THAN THE ALWAYSDISPLAYEDAREA.

The HardBorder prefab

Add the HardBorder prefab to a scene to set up a solid border around the edge of the screen or the AlwaysDisplayedArea. This border acts like a wall, and physics-enabled objects will collide with it.

The Sports demo scene has an example of the HardBorder prefab.

The Demos

We've included several sample scenes that demonstrate the key functionality.

- A Sports scene which demonstrates the Canvas and Background for a static game screen in landscape mode
- A Space scene which demonstrates camera locking to ensure no 'black bars' appear in your game.
- A side-scrolling scene that demonstrates how to move the camera while ensuring it only shows what you want it to.

Each demo scene has a short instruction file included.

USE UNITY'S FREE RESOLUTION SETTING AND DETACH THE GAME WINDOW SO YOU CAN CHANGE THE RESOLUTION ON THE FLY IN THE UNITY EDITOR TO SEE THE ASSET IN ACTION

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:

- The ResolutionManager script
- The AlwaysDisplayedArea object
- The MaximumBounds object.

The ResolutionManager script

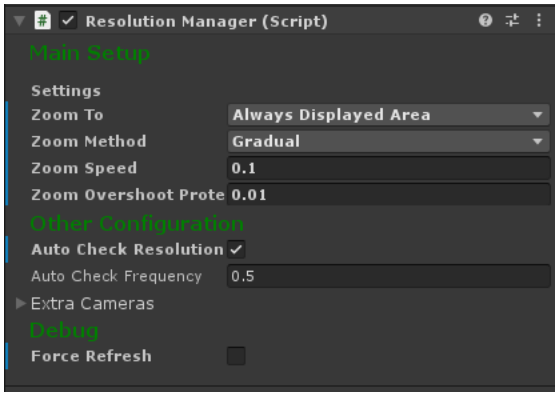
ResolutionManager has three main components:

- User settings
- Automatic functions
- Public methods you call from your own code.

User settings

Modify Resolution Manager's script settings to get the exact behavior you want. You can access all of these through the inspector by selecting the script instance in the game hierarchy.

The settings are intended to be set in the inspector and generally shouldn't be changed during runtime.



THE INSPECTOR SETTINGS FOR THE RESOLUTION MANAGER SCRIPT

Zoom To

Resolution Magic's main function is to zoom the camera to the best size based on your settings. You have two zooming options, *AlwaysDisplayedArea* and *MaximumBounds:*

- *AlwaysDisplayedArea* – make the canvas area fill as much of the screen as possible (while ensuring the entire area is visible).
    - The area will fit the screen height and/or width fully regardless of the screen ratio, and any extra space is filled with your MaximumBounds (where available).
- *MaximumBounds* – show as much of the background as possible without showing content outside of the background area (if possible).
    - The canvas will be completely visible in the centre of the screen, but there will be background content visible around the canvas most of the time.
    - Recommended for larger screens/high resolution screens, as it shows as much content as possible.

Default: *AlwaysDisplayedArea*

Zoom Method

(new in v3.0) You can choose between changing the camera zoom instantly or by a gradual zooming in or out.

If you choose the *Gradual* option, you can also customize how quickly the camera zooms to the correct scaling. This number represents how far the camera will zoom each frame until it reaches the desired size. If you set this too high, the zoom may 'overshoot' the desired scale. Smaller numbers will scale more slowly, but will be more accurate.

Zoom Overshoot Protection Speed

(new in v3.0) When the camera is zoomed to the correct scale, it will usually scale slightly past where you want it, and Resolution Magic will automatically scale the camera back to the correct size using this increment. The smaller this number is the more accurately the camera will be zoomed.

Auto Check Resolution Change

Resolution Magic periodically (see **Auto Check Frequency** below) checks if the screen resolution has changed, and resets the screen. This should be used on platforms and devices where the player can change the resolution and/or ratio during gameplay (e.g. Windows Store apps, or when your game supports landscape and portrait).

Default: *enabled*

Auto Check Frequency

How often the automatic resolution check occurs. You should normally leave the default value.

The value is the time between checks, so a lower number leads to faster checks.

Default: *0.5 seconds*

Restrict Screen to Background

When this is selected any camera movement *initiated by Resolution Magic* will only happen if the movement does not result in areas outside the background being displayed. It effectively locks the screen into only showing the Background and Canvas areas.

This doesn't affect camera movements from elsewhere in your code, only when you move the camera via the ResolutionManager script.

YOU CAN SEE THIS BEHAVIOR IN THE SPACE SAMPLE SCENE. MOVE THE CAMERA WITH THE ARROW KEYS. WITH THIS OPTION SELECTED YOU CAN'T MOVE THE CAMERA FAR ENOUGH TO SHOW ANYTHING OUTSIDE THE ORANGE BACKGROUND. DESELECT THIS OPTION AND YOU CAN MOVE THE CAMERA ANYWHERE, AND YOU WILL SEE THE BRIGHT PINK AREA BEYOND THE BACKGROUND IF YOU MOVE TOO FAR.

A typical example would be a platform game where the camera follows the player. If the player jumps near the top of the screen you won't want the camera to move up too far and expose blank areas.

Default: *enabled*

Force Refresh

For testing/debugging only. Tick this during testing to force the resolution to update. The box will instantly untick after refreshing the resolution.

CameraZoomChanged Event

This event is raised whenever Resolution Manager changes the camera size. It notifies any subscribers of the previous camera size and the new size.

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 Awake() or 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);
```

Canvas 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.

- CanvasLeftEdge
- CanvasRightEdge

- CanvasTopEdge
- CanvasBottomEdge

Multiple cameras

You can add extra cameras to the `Camera[] ExtraCameras;` property. These cameras will automatically be modified to match the main camera. An example of using multiple cameras is when you use a separate camera to show UI content or overlays and you want the scaling to match the rest of your game.

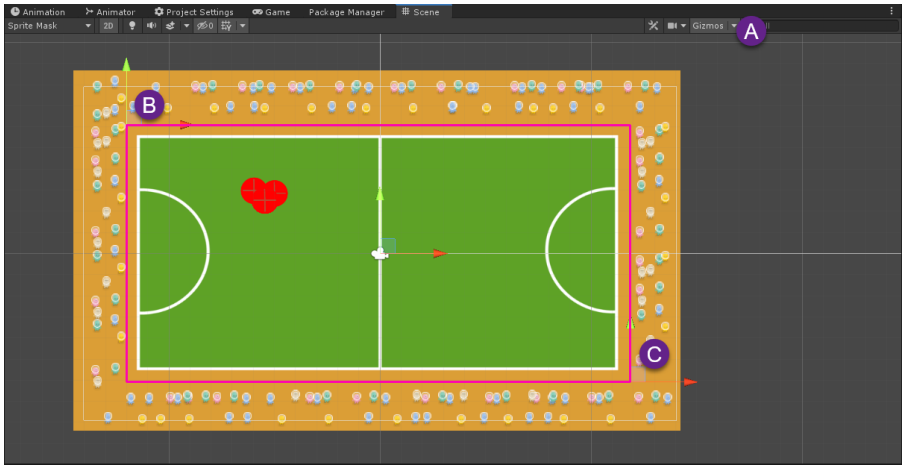
Drag extra cameras into the ExtraCameras array in the inspector to make ResolutionManager modify those cameras to match the main one.

The AlwaysDisplayedArea

Use the AlwaysDisplayedArea 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 to enable reshaping the area
B) There is one handle for the top left corner
C) And one handle for the bottom right.



EVERYTHING INSIDE THAT PINK RECTANGLE WILL BE VISIBLE ON ANY SCREEN THIS GAME IS PLAYED ON

Some content outside the designated area may display depending on the screen ratio and your settings.

Resolution Magic2D zooms the camera to a point that ensures the entire AlwaysDisplayedArea is visible. Depending on your settings the camera may show more screen area when possible.

AlwaysDisplayedArea Quick Notes

- All content within the AlwaysDisplayedArea will be visible to all players
- Resolution Magic will centre the screen on the AlwaysDisplayedArea 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
- By default, UI buttons within the canvas area will be moved to the edges of the screen
- Turn the canvas's sprite renderer on to see a translucent overlay and box to help you visualize your game area and place UI buttons

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.

The 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 AlwaysDisplayedArea 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.

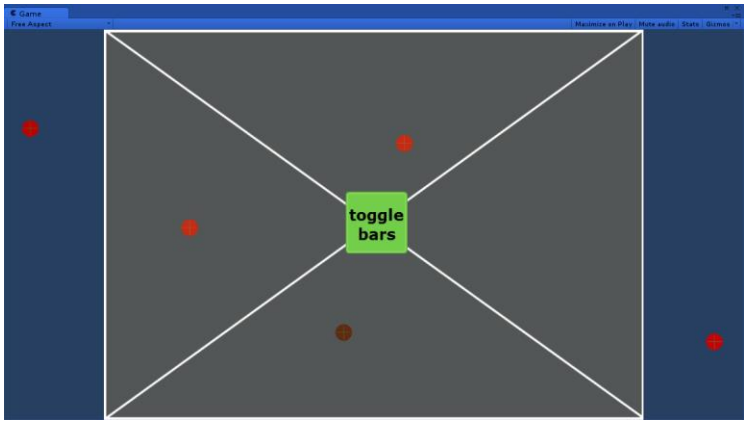
Black Bars

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).

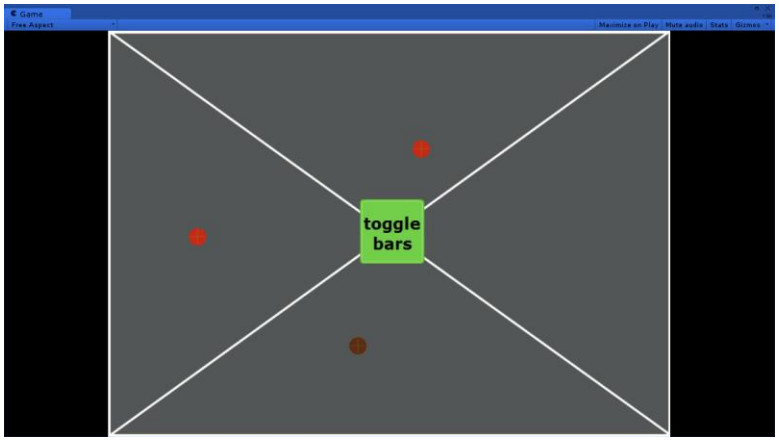
This should only be used if your players absolutely have to all see the same screen region (i.e. the AlwaysDisplayedArea) and you don't implement any way to prevent extra content showing in extra screen space available on wider/taller devices.

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

Example



A SCREEN WITH NO BLACK BARS. THE UNITY DEFAULT BLUE BACKGROUND IS VISIBLE OUTSIDE THE CANVAS AREA. NOTE THE TWO RED CIRCLES VISIBLE OUTSIDE THE CANVAS. PLAYERS WHOSE SCREEN IS THE SAME SIZE AS THE CANVAS WILL NOT SEE THESE CIRCLES BUT PLAYERS WITH A WIDER SCREEN WILL.



WHEN BLACK BARS ARE USED ALL PLAYERS WILL ONLY SEE THE CANVAS AREA REGARDLESS OF THEIR SCREEN SHAPE.

Use with care

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 Scenarios

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 periodically check for any change in screen size or ratio and adapt if there is a change.

- Put a copy of the Resolution Magic 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, enable Auto Check Resolution Change.

Every Player Should See Exactly the Same Content – Never More

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 AlwaysDisplayedArea.

- Setup Resolution Magic as usual

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 AlwaysDisplayedArea.
- You can force a refresh of the camera size by calling

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

- Once the resolution has been set to match your AlwaysDisplayedArea, 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

Move the camera

```
public void MoveCameraDirection(CameraDirections direction, float moveDistance, bool moveSafely = 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.

Example

```
// move the camera 0.5f to the left, but only if doing so doesn't reveal
area outside my game design

ResolutionManager.Instance.MoveCamera(CameraDirections.left, 0.5f, true);
```

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

This will move the camera directly to the given newPosition. If moveSafely is true, the camera will only move if the movement keeps the camera's view within the MaximumBounds. If moveSafely is false, the camera will be moved regardless of whether the movement reveals content outside MaximumBounds.

Refresh the resolution

```
public void RefreshResolution()
```

Forces a resolution check, which then triggers the camera to adjust and th UI buttons to re-align. This is done automatically by default, so there is no need to call it in normal circumstances.

Example

```
// 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 script

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
```

Example

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 3.0 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.
- (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.