Resolution Magic 2D

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
- Intelligently keeps UI buttons at the screen edges on all devices
- Adjusts to changes in orientation and resolution during the game
- Works on all platforms and doesn't interfere with other Unity features like the UI features added in Unity 4.6.

And:

- NO MORE black bars
- 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:

Canvas

The part of your game area that is always displayed regardless of the screen.

Background

The entire area that includes game content, including extra content that may display depending on the screen ratio.

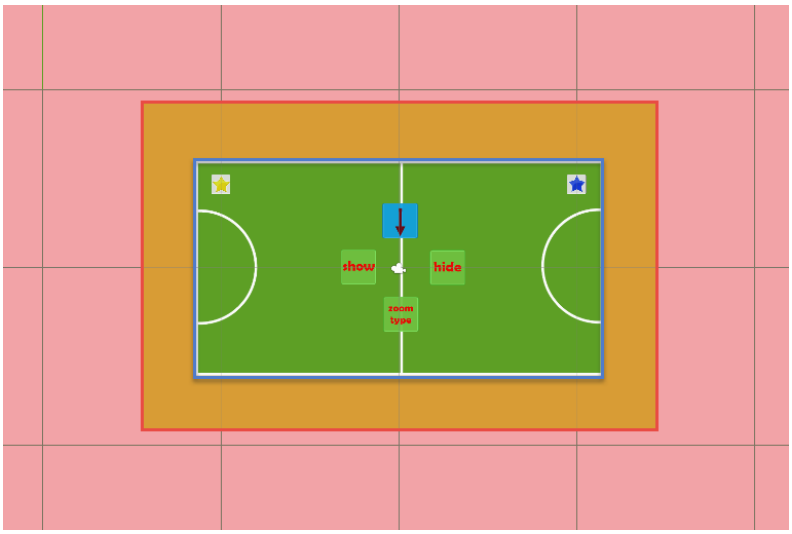


FIGURE 1BACKGROUND AND CANVAS

In Figure 1 there are three distinct areas of a sports game screen. The green area (canvas) displays 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 Canvas region, and use the Background 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.

Does it do anything else?

Yes! Resolution Magic 2D can manage your UI positions for you. Tell it where your button belongs (e.g. the top-left corner), and Resolution Magic 2D will ensure it is in the top-left corner on every screen your game is played on.

Here's an example of the same UI on different screens:

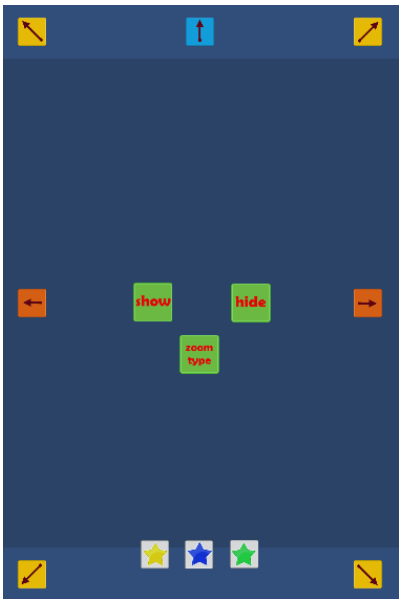


FIGURE 2-BUTTONS ON A PORTRAIT SCREEN

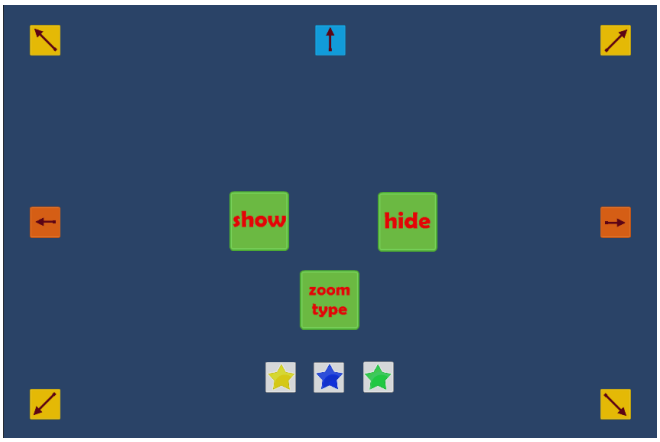


FIGURE 3 - BUTTONS ON A LANDSCAPE SCREEN

The UI items stay snapped to the edges of the screen regardless of resolution and ratio, and it's all automatic!

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 canvas/background rectangle sizes and choose where your buttons should go. There's are detailed instructions and three sample scenes to get you started. If you ARE a Unity guru you can follow the Quick Start Guide.

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

Advanced Unity gurus (and impatient people) 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 covers how everything works (without any code).

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 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 canvas and background sizes to suit your game's content

To use the UI features (OPTIONAL)

1. Add the AlignedObject script to any GUI items you want to align with the screen
2. Place your GUI items within the canvas area and set their properties to align to the relevant edge

TURN ON THE CANVAS'S SPRITE RENDERER WHILE PLACING OBJECTS SO YOU CAN SEE HOW THEY WILL ALIGN

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 IT SHOULDN'T TAKE MORE THAN A FEW MINUTES TO ADD IT TO AN EXISTING PROJECT.

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.

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:

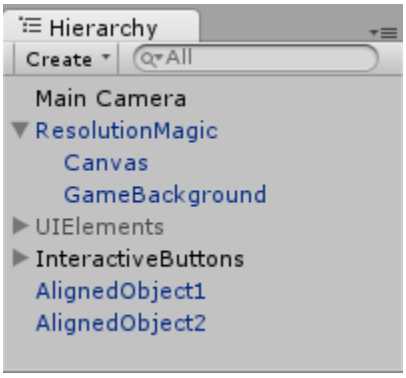


FIGURE 4- THE RESOLUTION MAGIC PREFAB IN THE HIERARCHY

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
- Shows and hides UI elements and solid border.

Methods you call from your code

- Show or hide UI objects
- Force a resolution check/change (you normally wouldn't call this manually, but you can).

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 Canvas

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

Canvas

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

Think of the Canvas as a template for your game screen. It includes a guide overlay that shows you what area will always be visible to your players. On screens that are a different ratio to the canvas, any content outside of the Canvas will display in addition to the Canvas view area.

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

THE CANVAS MUST STAY ACTIVE IN YOUR SCENE, BUT DESELECT ITS SPRITE RENDERER IN THE INSPECTOR TO HIDE IT WHEN YOU ARE NOT DESIGNING YOUR LAYOUT.

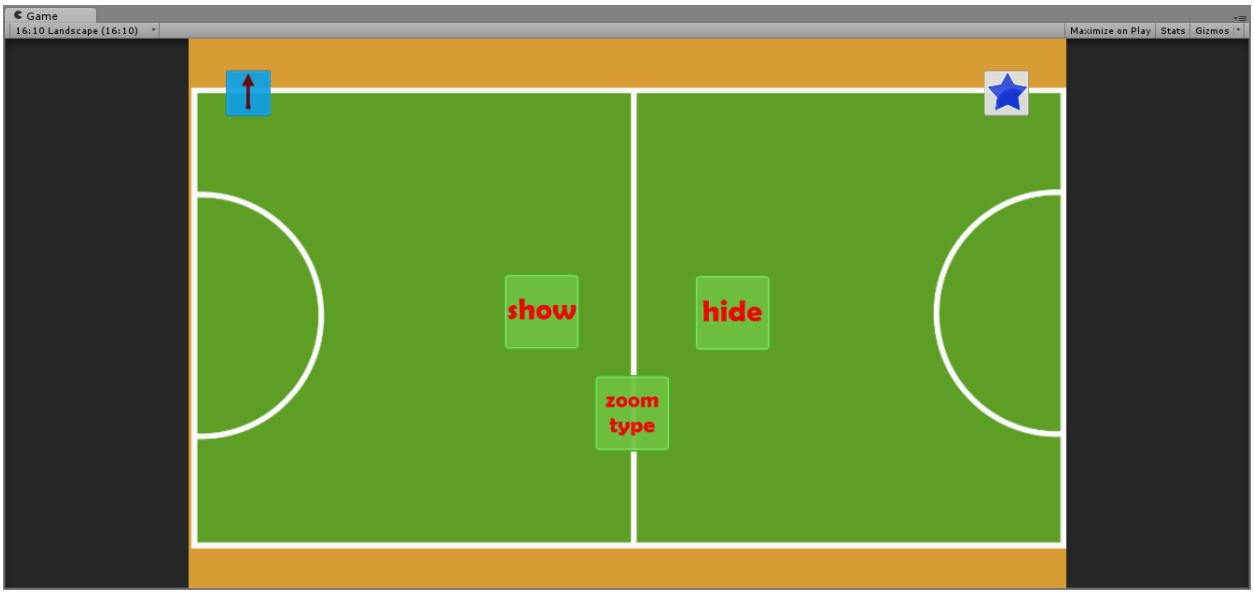


FIGURE 5- THE GREEN SPORTS COURT FILLS THE CANVAS SO IT IS ALWAYS VISIBLE, THE ORANGE BACKGROUND IS PARTIALLY VISIBLE BECAUSE THE CANVAS IS WIDER THAN THE SCREEN.

Background

The Background 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 canvas ratio. This usually means you should have a 'buffer' around your canvas with 'optional' content.

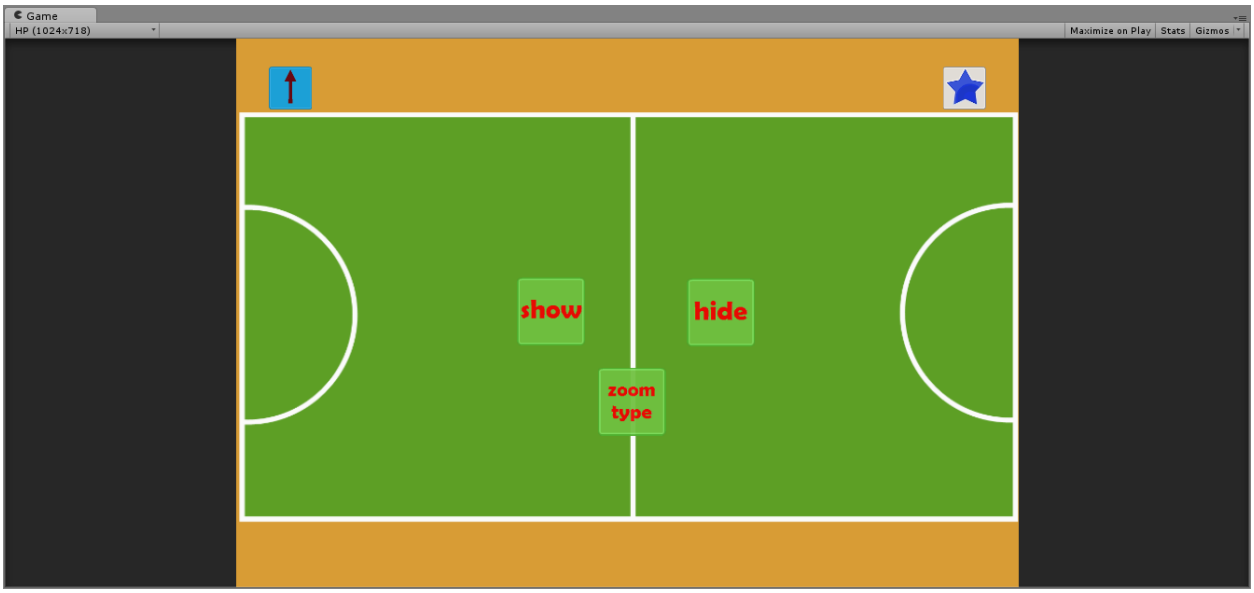


FIGURE 6 - ON A DIFFERENT SCREEN RATIO THE BACKGROUND IS MORE VISIBLE. NOTE THE UI BUTTONS HAVE MOVED TO STAY NEAR THE SCREEN EDGES. THE CANVAS STILL TAKES UP AS MUCH OF THE SCREEN AS POSSIBLE.

THE BACKGROUND MUST BE BIG ENOUGH FOR THE ENTIRE CANVAS TO FIT INSIDE.

The AlignedObject script

The AlignedObject script can be added to any game object to ensure it stays in its position relative to the screen edge. Resolution Magic 2D can also move these objects off and on the screen to hide/show the UI.

The HardBorder prefab

Add the HardBorder prefab to a scene to set up a solid border around the edge of the screen or Canvas area. 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 three sample scenes that demonstrate the key functionality.

- A UI scene, which demonstrates UI alignment
- 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.

Each demo scene has a short instruction file included.

USE THE FREE RESOLUTION SETTING AND DETACH THE GAME WINDOW SO YOU CAN CHANGE THE RESOLUTION ON THE FLY IN THE UNITY EDITOR.

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 Canvas
- The Background.

The ResolutionManager script

ResolutionManager has three main components:

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

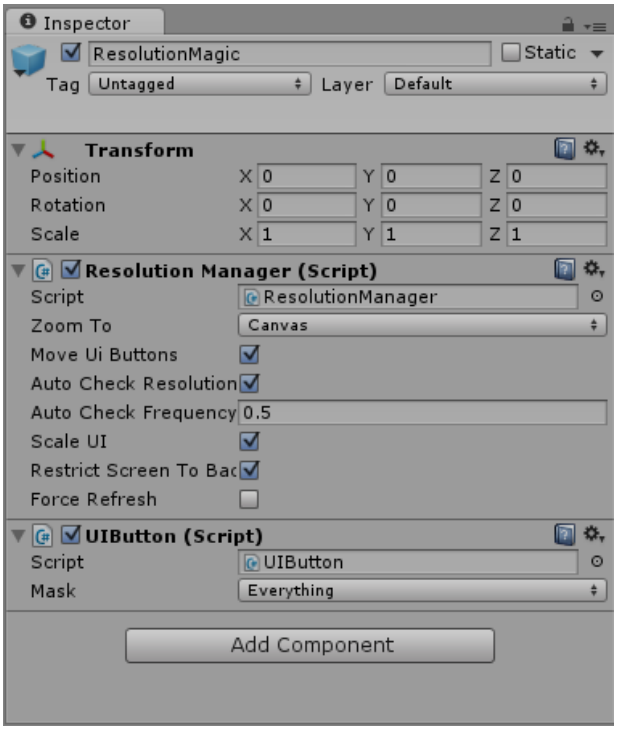
User settings

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.

You can access these settings via code if you want, e.g.:

```
ResolutionManager.Instance.ScaleUI = false;
```

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



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, *Canvas* and *Max Size*:

- *Canvas* – make the canvas area fill as much of the screen as possible (while ensuring the entire canvas area is visible).
    - The canvas will fit the screen height and/or width fully regardless of the screen ratio, and any extra space is filled with your Background (where available).
    - The Canvas is NOT stretched.
- *Max Size* – 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: *Canvas*

Move UI Buttons

When selected, the ResolutionManager script will move anything with the AlignedObject script attached off and on the screen when the PlaceUI method is called (you can override this behavior for individual items – see the AlignedObject section for details).

Default: *enabled*

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*

Scale UI

Resolution Magic can scale UI items to match the screen. UI items will appear in proportion to their size in the editor regardless of your screen size.

THE SCALING IS APPROXIMATE, BUT IS FINE FOR MOST CASES.

Default: *enabled*

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.

Automated tasks

The following are automatic, and require no code or intervention from you.

Camera adjustment

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 from landscape to portrait.

Aligning objects

Any objects that have the AlignedObject script attached are placed automatically in the correct location whenever the camera is adjusted. Offscreen positions are also calculated so objects can be moved off the screen when requested. This behaviour can be disabled from the script settings.

Public methods

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

Show and hide the UI

Any UI buttons that have the AlignedObject script attached will be hidden and shown on the game screen according to their properties. ResolutionManager will handle the button positioning based on the canvas and screen sizes/edges.

Examples:

```
ResolutionManager.Instance.ShowUI(); // show UI items on the screen

ResolutionManager.Instance.HideUI(); // hide UI items off the screen
```

Move the camera

You can request that ResolutionManager move the camera in a specific direction and distance, and also specify if you want the move to be *safe*. When the camera is moved safely, it will only move if doing so does not reveal areas outside of the Background to the player. Unsafe movement will move the camera regardless.

If you want to always move the camera safely, enable the Restrict Screen to Background property. If you want to sometimes move safely and sometimes unsafely, leave that property disabled and use the moveSafely parameter (which is optional and defaults to false when not used):

```
public void MoveCamera(CameraDirections direction, float moveDistance,
bool moveSafely);
```

SETTING *RESTRICT SCREEN TO BACKGROUND* OVERRIDES THE *MOVESAFELY* PARAMETER MAKING IT ALWAYS TRUE!

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 in other areas of your game. These are used to find the positions of the screen edges, and can aid in placing game items or UI items.

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.

- 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 Canvas

Use the canvas to define your core game screen area. Anything within its bounds is guaranteed to be displayed on any screen your game is played on.

If you want Resolution Magic 2D to control your UI button placement, place your buttons within the canvas area. Resolution Magic 2D will note their location relative to the edge(s) of the canvas space and then re-align the buttons to the screen edges if required.

The canvas has a sprite renderer that displays a semi-transparent overlay when active. Use this to guide your game layout, but remember to disable it when not using it.

In a nutshell

Content within the canvas area will always display on the player's screen

Content outside the canvas area may display depending on the screen ratio and your settings

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

THE RESOLUTIONMANAGER SCRIPT RECOGNIZES THE CANVAS BY ITS TAG 'RM_CANVAS', SO DON'T CHANGE THIS!

Using the canvas to place GUI elements

You probably want to make sure your user interface items align with the screen edges (e.g. if your canvas is 4:3 but your player has a 16:9 screen, you want the buttons on the edges of the screen (i.e. outside the canvas).

By default UI buttons 'stick' to the screen edge, e.g. if you place the button 50pixels from the top of the *canvas*, that button will always be 50 pixels from the top of *screen* at runtime, regardless of whether the canvas reaches the top of the screen.

Canvas notes

- All content within the canvas bounds will be visible to all players

- The canvas must be centred on the screen. Resolution Magic automatically achieves this by centering the camera on the canvas
- 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 Background

The Background defines the full content in your scene. It is defined by the 'RM_Background' tag. 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 canvas is fully zoomed. The background should contain content that the player doesn't NEED to see (as that should be in the canvas) but that the player *may* see depending on their screen.

Typically you would make your background quite a bit larger than your canvas so that there is content to display on the edges when your canvas is a different shape/ratio from the player's screen.

If you choose to zoom your game to display the most possible content, the camera will zoom out as far as it can while not showing anything outside the area defined by the background (if possible - you need to make sure your background is big enough to cater to any screen shape).

The prefab contains a default background, which you can easily change for your own background content as needed. The only parts that Resolution Magic cares about are the background 'RM_Background' tag and the area the background covers. The background needs to be bigger than the canvas.

The background doesn't actually need to hold an image, it just needs to be present to mark out the 'maximum' area of your game. Many of the ResolutionManager properties and methods rely on the background to work.

For games where the camera moves, you can use the Background 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.

UI Buttons

Resolution Magic contains some automated behaviour for user interface objects, typically used for buttons, such as a pause button in the corner of the screen or a health bar.

The default behaviour for UI buttons is to align them to the screen edge and to show and hide them by sliding them off and on screen when the Resolution Manager script's ShowUI() or HideUI() methods are called. You can modify or override the default behaviours to suit your game.

The AlignedObject script

The AlignedObject script must be attached to any UI object that you want Resolution Magic to automatically manage. You may then adjust the properties of the AlignedObject script on your UI object to suit your needs. The separate properties are explained below:

Aligned Edge

Each UI button object must be aligned to a screen edge (left, right, bottom, top-left corner, etc.). This is what ensures the button stays at the edge when the screen size or ratio changes. This works with Aligned To to determine the button's place on the screen.

Aligned To

By default, all UI is aligned to the screen, meaning that a button aligned to the upper-left corner will be placed in the upper-left corner of the player's screen regardless of the screen ratio and resolution. This prevents buttons from appearing in the middle of the game area when the screen shape differs from the canvas shape.

You can choose to align elements to the canvas instead, which keeps the UI within the pre-defined canvas area.

Move Speed

This determines how quickly the UI object moves off/on the screen when it is told to show or hide. The default is 0.5, which is a smooth slide. Lower values increase the speed. Use 0 to make the item show or hide instantly.

Ignore Manager

Enable this to make the button ignore commands to show and hide delivered by the Resolution Manager script. This is useful when you want to manage one or more buttons separately from the rest. For example you may want to have a pause button that stays on the screen regardless of what is happening with the rest of the UI.

Start Off Screen

When selected, the object will be hidden (using the HideInstant() method) at the start of the scene.

Note: You still need to place the object within the canvas area when designing the layout.

AlignedObject commands

You can call the following commands to any UI button containing the AlignedObject script from any other script in your game (i.e. they are public):

HideThis() and ShowThis()

Hides or shows the object, animating it according to its settings.

When IgnoreManager or DontMove is active these methods are ignored.

HideNow() and ShowNow()

These methods will hide or show the object regardless of its settings (i.e. the IgnoreManager and DontMove settings. The object animates into position according to its settings.

HideInstant() and ShowInstant()

These methods instantly show or hide the object. Animation is skipped, and the IgnoreManager and DontMove settings are ignored.

Bounce()

'Bounces' the button to simulate a button press. This shrinks and expands the button's attached sprite briefly, and should be called when your player presses the button.

The solid border

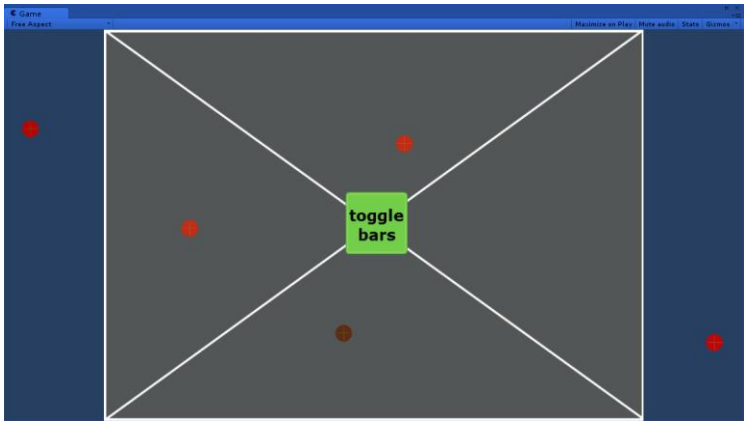
This is an optional feature. When enabled, a box is created around the edge of the screen or canvas (you can choose). This box is solid, in that it has a collider forming a wall on the screen edge that physics objects will collide with. You can set each edge to be on or off individually. The border is off by default and is enabled in the ResolutionManager script settings.

Black Bars

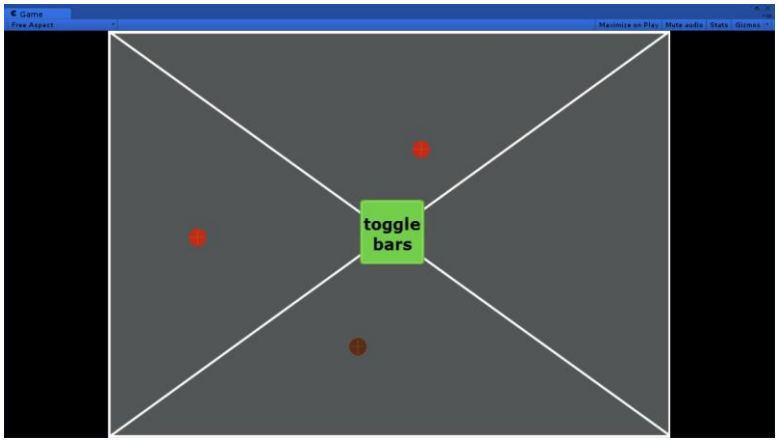
The Black Bars prefab can be added to the camera to fill any empty space in the screen with 'black bars' (like when you watch an old 4:3 TV show on a widescreen TV).

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

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.

Code reference

ResolutionManager

Move the camera

```
public void MoveCamera(CameraDirections direction, float moveDistance, bool moveSafely = false)
```

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 (defaults to false if not included), and when true the camera will only move if doing so does not reveal area outside of your game's background 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);
```

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();
```

Hide the UI

```
public void HideUI()
```

Moves items with the AlignedObject script attached off the screen according to their alignment settings.

Note: you can set individual items to ignore this command by setting the IgnoreManager property to true.

Example

```
// hide UI items off the screen

ResolutionManager.Instance.HideUI();
```

Show the UI

```
public void ShowUI()
```

Moves items with the AlignedObject script attached onto the screen according to their alignment settings.

Note: you can set individual items to ignore this command by setting the IgnoreManager property to true.

Example

```
// show UI items onto the screen

ResolutionManager.Instance.ShowUI();
```

Note: ResolutionManager also provides public access to many properties that you can use to find and calculate screen positions, such as finding the screen edges and corners. The ResolutionManager script contains comments for what each property returns.

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.

AlignedObject

Hide this object

```
public void HideThis()
```

Moves this object off the screen according to its alignment settings. This is also called when the HideUI method in the ResolutionManager script is run.

Example

```
alignedObject.HideThis();
```

Note: The when IgnoreManager or DontMove is enabled this method does nothing.

Show this object

```
public void ShowThis()
```

Moves this object onto the screen according to its alignment settings. This is also called when the ShowUI method in the ResolutionManager script is run.

Example

```
alignedObject.ShowThis();
```

Note: The when IgnoreManager or DontMove is enabled this method does nothing.

Hide this object now

```
public void HideNow()
```

Moves this object off the screen according to its alignment settings. This method ignores the IgnoreManager and DontHide settings, so it will ALWAYS hide the object when called.

Example

```
alignedObject.HideNow();
```

Show this object now

```
public void ShowNow()
```

Moves this object onto the screen according to its alignment settings. This method ignores the IgnoreManager and DontHide settings, so it will ALWAYS show the object when called.

Example

```
alignedObject.ShowNow();
```

Hide this object instantly

```
public void HideInstant()
```

Moves this object to its off-screen position instantly, without the animation. This method ignores the IgnoreManager and DontHide settings, so it will ALWAYS hide the object when called.

Example

```
alignedObject.HideInstant();
```

Show this object instantly

```
public void ShowInstant()
```

Moves this object to its on-screen position instantly, without the animation. This method ignores the IgnoreManager and DontHide settings, so it will ALWAYS show the object when called.

Example

```
alignedObject.ShowInstant();
```

Bounce this item

```
public void Bounce()
```

Causes the item to 'bounce' as if pressed like a button.

Example

```
alignedObject.Bounce();
```

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 also some extra generic scripts in the example scenes used for simple mouse input, and these are also not documented as part of Resolution Magic 2D.

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.

- 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.
- Make sure your prefab tags have not been changed:
    - Canvas: RM_Canvas
    - Background: RM_Background
- Re-import the asset into your game to reset everything.
- Make sure the scripts are attached where they need to be. UI objects need the AlignedObject script in order to be moved by the ResolutionManager script; 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 referncing 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.