madVR v0.89.5

information for media player and subtitle renderer developers

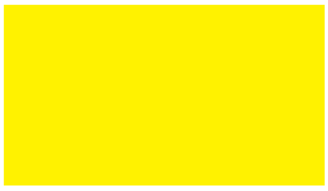
First of all, if your media player already supports older madVR builds, there’s nothing you *have* to change. Everything should continue to work fine with v0.89.5 and newer builds. However, if you want to provide your end users with the best possible experience, there are a few very simple changes you can implement to get the best out of madVR v0.89.5 (and newer).

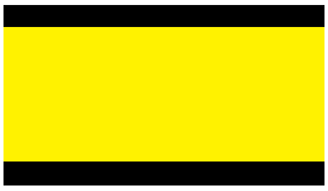
The new builds (v0.89.x) introduce a number of new features. In this document I’ll describe the purpose and behavior of these new features, and which changes you can do to optimize your media player for that.

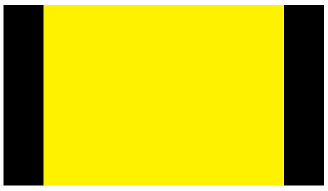
1) automatic detection of hard coded black bars

1a) problem description

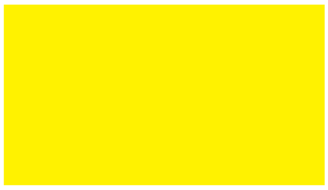
Blu-Ray movies are pretty much always encoded as 1920x1080, although most movies actually have a different native AR (aspect ratio). The majority of movies are either 2.40:1, 1.85:1 or 4:3. Practically that means that a great many Blu-Rays have hard coded black bars in the encoded image.

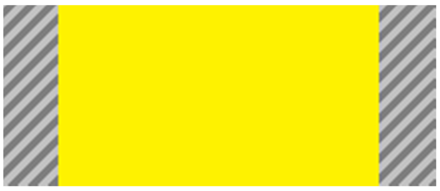

16:9

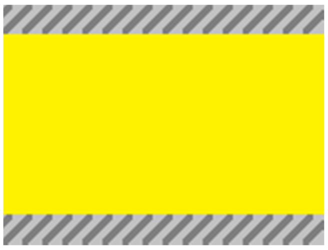

Cinemascope Blu-Ray


4:3 Blu-Ray

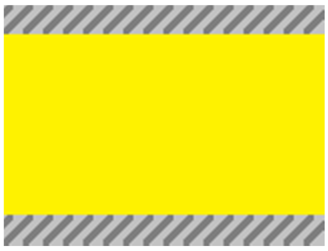
Most users have a 16:9 TV or projector. So Blu-Ray having a 16:9 encoding format is a good match. If the Blu-Ray is well encoded, users with a 16:9 TV will see all image content, and the black bars won't be larger than necessary. **However, what happens if the TV has a different aspect ratio?** Usually the media player doesn't know which native AR the movie has. So it will usually "fit" (position & scale) the image into the TV so that no image content is lost, but that there are some small black bars, which is just fine for native 16:9 content:

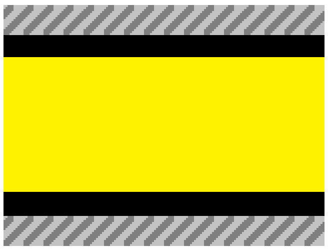

16:9 BR -> 16:9 TV

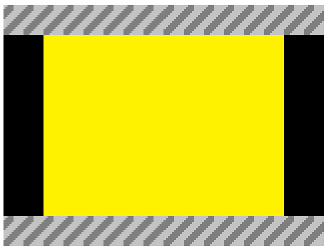

16:9 BR -> Scope TV

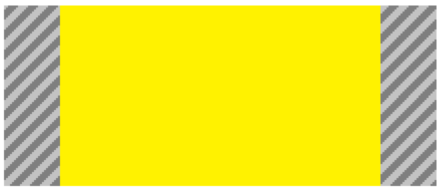

16:9 BR -> 4:3 TV

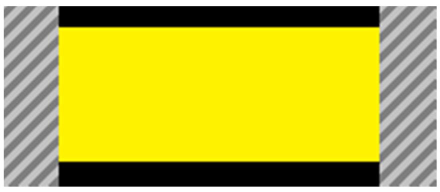
So far, so good, but now it gets tricky. As mentioned above, the media player usually doesn't know the native AR of the movie. So the scaling of the video will still be the same, even if the native AR of the movie is 4:3 or 2.40:1.

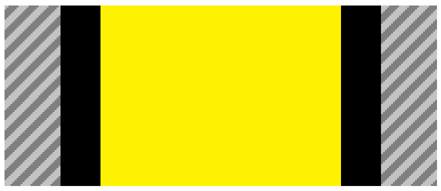

16:9 BR – 4:3 TV


Scope BR – 4:3 TV


4:3 BR – 4:3 TV


16:9 BR – Cinemacope TV


Scope BR – Scope TV


4:3 BR – Scope TV

As you can see, playing back a 4:3 Blu-Ray on a 4:3 TV, or playing back a Cinemascope Blu-Ray on a Cinemascope TV, usually results in heavy pillarboxing (meaning black bars all around the image). That is a very bad experience for the end user, who will usually not understand why the media player doesn't zoom the image "correctly".

Then there are also Blu-Rays which have hard coded pillarboxing, which usually the media player doesn't know about, either.

1b) feature description

Starting with v0.89.x, madVR now has the ability to automatically detect any hard coded black bars in the encoded video image. There are various options that allow the end user to fine tune how he wants scaling to be done exactly, how to handle if a movie changes AR in the middle of playback etc. You'll find all those options in the madVR settings dialog under "processing -> zoom control".

Since all those options are quite complex and may partially influence each other, I thought it would be best to "hide" all this complexity from the media player. So there's not much you need to do. However, this decision made it necessary for madVR to no longer strictly obey your zooming wishes. Instead madVR now tries to "understand" what you want, and then reinterprets that, based on the detection of the black bars, and on which exact zooming options the user has activated in madVR. However, your wishes can be ambiguous to madVR, because I only have your IBasicVideo::SetDestinationPosition() call to look at, and that's just not a lot of information to work with.

1c) changes you can do – part 1

If your media player doesn't support different zooming modes, you could simply remove any and all calls to IBasicVideo::SetDestinationPosition(). In that case madVR will default to "touchInside", which is probably the "best" zooming mode, anyway.

If you want more control, I've added new APIs through the "IMadVRCommand" interface, which you can use now instead of (or in addition to) IBasicVideo::SetDestinationPosition() to specify exactly which zooming and position wishes you have. Using the new APIs is recommended in most cases because it takes the "guessing" out of madVR's behavior.

Here's a list of available commands:

IMadVRCommand::SendCommandStr("setZoomMode", whatever);

Supported values are: "autoDetect", "touchInside", "touchOutside", "stretch", "100%", "50%", "200%" and many more percentage values. The default value is "autoDetect", which means that madVR tries to understand your IBasicVideo::SetDestinationPosition() call. If you set the zoom mode to any other value, madVR will totally ignore your calls to IBasicVideo::SetDestinationPosition().

IMadVRCommand::SendCommandDouble("setZoomAlignX" / "setZoomAlignY", whatever);

Allows you to align the video rectangle "left/top", "center" or "right/bottom". Center is default.

IMadVRCommand::SendCommandDouble("setZoomFactorX/Y" / "setZoomOffsetX/Y", whatever);

Allows you to specify an additional zoom which is applied (via multiplication) on top of the zoom mode. Or an additional offset to the video rectangle, on top of the chosen alignment mode.

1d) changes you can do – part 2

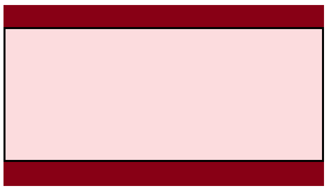
There are some movies which change AR in the middle of playback. Good examples are IMAX Blu-Rays like "The Dark Knight" or "Interstellar". If the user has activated the option "zoom control -> notify media player about cropped black bars" madVR will send an EC_VIDEO_SIZE_CHANGED notification to you, every time the AR of the movie changes.

It might make sense for you to double check that your media player handles all of this correctly. Also, if your media player adjusts the window size to the video size, you may also want to make sure the window size is always updated when madVR sends such an EC_VIDEO_SIZE_CHANGED notification. You can ask IBasicVideo::GetVideoSize() to get the newly detected active video size.

2) screen masking

2a) problem description

Pretty much every home theater optimized front projector has a 16:9 sized panel and thus throws a 16:9 image and is seen by Windows as a 16:9 TV. However, many front projection users have screens with different aspect ratios. For the sake of this document let's just talk about one specific case: Let's say the user has a 16:9 projector, but a Cinemascope (2.40:1) screen. This just a rather common situation. There are various ways to handle such a setup. To make it simple, let's presume the user has setup the projector in such a way that the 16:9 projected image covers the whole screen. Obviously that means that some part of the 16:9 panel is projecting outside of the screen:


dark red stripes = projected image outside of cinemascope screen

By default, media players will in this situation zoom 16:9 Blu-Rays so that the whole 16:9 projector panel is filled. That happens to be fine for Cinemascope movies, because the encoded black bars in the 16:9 Blu-Ray should black out the parts of the projector panel which are outside of the screen, anyway. However, if you playback an actual 16:9 movie, parts of the active video area will be painted outside of the screen, which obviously is very bad.

2b) feature description

Starting with v0.89.x, if the user defines his display device in the madVR settings to be a "Digital Projector" or "CRT Projector", there will be an additional "screen config" settings page in the device setup. This new settings page will allow the user to "mask off" some parts of the 16:9 projector panel to redefine the native size and AR of the screen.

This masking is then later used by madVR, but only if the media player is in fullscreen mode (doesn't matter if it's exclusive or not). Masking is ignored in windowed mode. The masking is used for 2 purposes:

1) The active video rectangle is automatically repositioned and zoomed by madVR to fit into the visible screen area.
2) OSDs and subtitles are moved into the visible screen area, if madVR has an influence on that.

2c) suggested test setup

For your tests, I'd suggest that you define your display in the "device" section of the madVR settings dialog to be a "Digital Projector". Then in the "screen config" page activate the "define visible screen area" option and choose reasonable values. E.g. for a 1920x1080 projector and a Cinemascope screen the visible screen area should be around 1920x800 pixels, so masking 140 pixels off top and bottom would make sense as a test case.

2d) changes you can do – when you're using ISubRenderCallback(2)

The ISubRenderCallback::Render() methods have parameters for the active video area rectangle. But they only have a *size* information for the rendering window/screen, not a rectangle. So I can't transport the masking information to your ISubRenderCallback through the interface.

Please use IMadVRInfo:: GetRect(„fullscreenRect") to get the exact rectangle into which you can render. Please note that "top" and "left" can be non-zero. You should position the subtitles either into the active video rect which you get as "Render()" parameters. Or to the "fullscreenRect". If you do that, your subtitles should always be inside of the visible projection screen area.

2e) changes you can do – when you’re using IMadVROsdServices::OsdSetRenderCallback

In older madVR builds the “fullOutputRect” parameter to your “ClearBackground” and “RenderOsd” callbacks always had “left” and “top” zeroed out. This is no longer the case if the left/top borders are masked off. So please double check your code that you take non-zero “fullOutputRect->left/top” values properly into account. That’s all!

2f) changes you can do – when you’re using IMadVROsdServices:: OsdSetBitmap

If you don’t change anything, madVR will automatically scale all your images down to the reduced visible screen area, and reposition them appropriately. Furthermore, all mouse position information will be translated for you automatically. So in theory you don’t need to change anything, everything should continue to work as intended.

However, your OSD will be scaled down using ugly Bilinear interpolation, which will result in a loss of sharpness, and may make small fonts much harder to read. Also, the aspect ratio of your OSD will likely not match the aspect ratio of the masked down screen area. madVR will preserve the AR of your OSD, so it won’t get distorted, but that also means that your OSD may not cover the whole visible screen area, anymore. If you want to do a fullscreen OSD, that might bother you.

I’ve made it very easy for you to optimize your OSD for the new masking feature. There are only 2 things you need to do:

1) Instead of sizing your OSD by using the window/screen size, ask madVR about the size of the visible screen area by calling IMadVROsdServices::OsdGetVideoRects() or IMadVRInfo::GetRect(“fullscreenRect”). Only use the width (= right - left) and height information you get from madVR. Don’t change your positioning. Position 0,0 is still the left/top corner of the visible screen area. The only thing you need to change is the width and height of your OSD, and only if you draw a fullscreen OSD. If your OSD isn’t fullscreen, you probably don’t have to change anything at all here.

2) In your IMadVROsdServices::OsdSetBitmap() call, add the BITMAP_MASKING_AWARE flag. This will tell madVR that you’ve taken screen masking properly into account, so that madVR doesn’t need to scale your OSD images.

Done!

If you have any questions, let me know. I can also provide samples with changing aspect ratios, if that helps.

madshi@gmail.com