I remember taking a screen shot of a video, and when I opened it in Paint, the video was playing in it! What witchcraft is this?

October 14th, 2025 5 reactions

# I remember taking a screen shot of a video, and when I opened it in Paint, the video was playing in it! What witchcraft is this?

Raymond Chen

Read next

September 30, 2025

**Microspeak: Convicted**

Raymond Chen

September 25, 2025

**Samples note: Use comments to describe what code does, not what you wish the code would do**

Raymond Chen

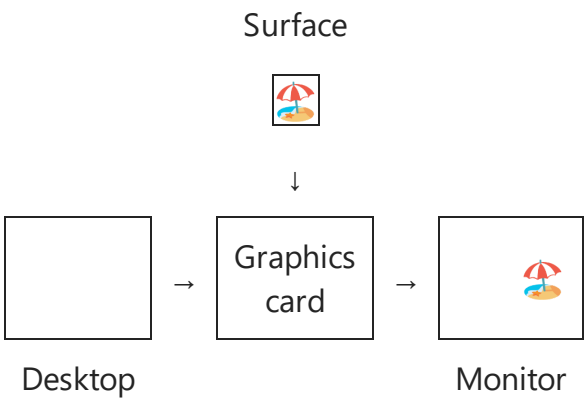
@ChenCravat In an old version of Windows (Windows 98 iirc) if you took screenshot of a video from media player and paste it into paint, and resume media player, video would play inside paint. Do you why it happened? It is still bugging me to this day.

— Yasar Arabaci @ysar.bsky.social (@y_arabaci) July 18, 2025

One of the tricks for video playback is to use a green screen, more technically known as color-keying or chroma-keying.

The media player program didn't render the video pixels to the screen. Rather, it followed this recipe:

- Draw solid green where you want the video to go.
- Render the video pixels to a graphics surface shared with the graphics card.
- Tell the graphics card that whenever it sees a green pixel about to be written to the screen, it should substitute a pixel from that shared graphics surface.




There are a few advantages to this approach.

One is that the shared graphics surface need not have the same pixel format as the user's main display. Therefore, you can specify that the shared graphics surface have a pixel format that matches that of the video, avoiding the need to do any pixel format conversions.

Another is that you can update the content without having to go through a full paint cycle. You just update the shared graphics surface, and the results are on the screen at the next frame. This lets you update the video at 60 frames per second from a background thread, which works even if the UI thread is busy or sluggish.

You can do even better if you create *two* shared graphics surfaces. The first one holds the contents of the video frame you want the user to see right now. And the second one is where you create the contents of the video frame you want the user to see next. And then at the vertical blank, you tell the video card to switch to the second shared graphics surface (known as "flipping"), and the entire screen updates at once with no tearing. While the second surface is on the screen, you can render the next frame to the first surface, and then flip again at the next vertical blank. Repeat this process for each frame of the video.

A media player program of this era typically negotiated with the graphics card (via DirectDraw) to get one of these magic graphic surfaces and configure it to use it as replacement pixels. These special surfaces were called "overlays" because they appeared to overlay the desktop.

When you took a screen shot, you got the pixels that Windows gave to the video card as the contents of the desktop. If an overlay is active, then these are not the same pixels that came out of the video card and sent to your monitor. The computer never sees these monitor pixels; they are something generated on the fly by the graphics card and sent directly to the monitor. Your screen shot was a screen shot of the desktop screen, and it contains green pixels where the video would go.

Now, when you load the image into Paint or any other image viewer, Windows sends those green pixels to the video card, but if the media player is still running, then its overlay is still active, and if you put Paint in the same place that the media player window is, then the green pixels in Paint get changed into the pixels of the active video. The video card doesn't know that the pixels came from Paint. Its job is to look for green pixels in a certain region of the screen and change them into the pixels from the shared surface.

If you move the Paint window to another position where it doesn't overlap the media player, or if the media player isn't playing a video, you will see the bitmap's true nature: It's just a bunch of green pixels.

Let's go back to the green screen analogy: Imagine you are visiting a television studio while the presenter is giving a weather report. The presenter is standing in front of a green screen, but the image that goes out to viewers contains an animating weather map where the green backdrop would normally appear. You take a picture of the green screen with your phone, and you hold up the phone to the television camera. What do the viewers at home see? Do they see a phone with a picture of a green screen? No, they see a phone with an animating weather map! And if you move your phone around, what the viewers at home see is different parts of the weather map being inserted into your phone. When you get home, your friends tell you, "Wow, how did you do that? You took a still picture of a weather map, and when you held it up, it was animating!"

Now, while overlays are better than going through paint cycles, they still have their problems. For example, if a window moves over the media player, and it happens to have green pixels, then the video will play in that other window. If you move the media player window, it needs to move the overlay to match the media player's new location, and in practice there is some lag to this tracking, causing it to look jerky. Also, there is a limit on the number of overlays supported by a graphics card, so if they're all used up, then the media player has to go through the old software rendering path.

Nowadays, video rendering is no longer done with overlays. Instead, content is rendered to graphic surfaces that are associated with window. The desktop compositors takes the graphics content of all the windows, including their composition visuals, and combines them to form a full desktop image that is sent to the monitor. The desktop compositor understands window positions, so when you move the window, the composition visuals automatically move with them, so you don't get the phenomenon of the overlay lagging the window position. The desktop compositor also understands visual transformations, so that when you hit <kbd>Alt</kbd>+<kbd>Tab</kbd> or hover over the taskbar button, the animating video is automatically resized and repositioned to match the preview thumbnail.

5 11 1

## Category

Old New Thing

## Topics

Other

## Share

## Author

Raymond Chen

Raymond has been involved in the evolution of Windows for more than 30 years. In 2003, he began a Web site known as The Old New Thing which has grown in popularity far beyond his wildest imagination, a development which still gives him the heebie-jeebies. The Web site spawned a book, coincidentally also titled The Old New Thing (Addison Wesley 2007). He occasionally appears on the Windows Dev Docs Twitter account to tell stories which convey no useful information.

## 11 comments

Join the discussion.

Sign in

Sort by : Newest

alan robinson October 15, 2025 0

I recall this from the Win98 era, maybe later.

Are you sure it was a green key? I recall the screenshot giving some kind of generic gray color not anything so saturated as full green. Or was that just an implementation detail depending on the video card driver/settings?

Log in to Vote or Reply

**James Lin** October 16, 2025 0

The overlays that I usually dealt with all used bright magenta (0xFF00FF).

Log in to Vote or Reply

**IS4** October 15, 2025 0

For me it was a very dark purple or something like that. Definitely not green since that color was likely to be used for normal purposes.

Log in to Vote or Reply

**alan robinson** 2 days ago 0

yeah that makes sense. I'm surprised a single color was good enough to key on, maybe there was an extra criteria like run length? I seem to remember it was just pixel color, in isolation or not, though.

**Alex** 3 days ago 0

> Nowadays, video rendering is no longer done with overlays.

It is still done that way on Android, at least on some devices.

Log in to Vote or Reply

**Paul** 2 days ago 0

Raymond is referring to Windows.

Log in to Vote or Reply

**Tim Dawson** 3 days ago 0

What an interesting article. I had always assumed this was what was happening, based on my own observations, but it's great to have it fully explained. I would love to learn more about how this all works nowadays, including more detail on the desktop compositor. Does it have fallbacks? Is the non-composited version of all this still around?

Log in to Vote or Reply

**Jacob Manaker** October 14, 2025 0

What happened when the user clicked "save" in Paint? Did they get an all-green BMP?

Log in to Vote or Reply

**Raymond Chen** Author October 15, 2025 0

What happens if you click "Save" in the Photos app on your phone after you took a picture of a green screen in a television studio?

Log in to Vote or Reply

**magnum2227 magnum2227** October 14, 2025 0

Hardware overlays, as well as showing videos in YUV bitmap format not RGB like the screen, would also smoothly resize the image if you made the playback window bigger or full-screen, much faster than the CPU could at the time.

Log in to Vote or Reply

**GL** 4 days ago 1

My last rememberance of such effect was in Windows XP. At that time I had vague understanding that video cards had some gimmicks for performance, but I was naive enough to think the pixels were special, not in color, but with other flags, because I remember the upper-left pixel would always be the upper-left pixel of the video — but this could be a memory error of mine. Later I called this technique "set transparent color" or "punctured neighborhood". 🤣

Log in to Vote or Reply

Get notified when new posts are published.

Enter your email

Subscribe

By subscribing you agree to our Terms of Use and Privacy

## Follow this blog