



**WIKIPEDIA**  
The Free Encyclopedia

[Main page](#)

[Contents](#)

[Featured content](#)

[Current events](#)

[Random article](#)

[Donate to Wikipedia](#)

[Wikipedia store](#)

Interaction

[Help](#)

[About Wikipedia](#)

[Community portal](#)

[Recent changes](#)

[Contact page](#)

Tools

[What links here](#)

[Related changes](#)

[Upload file](#)

[Special pages](#)

[Permanent link](#)

[Page information](#)

[Wikidata item](#)

[Cite this page](#)

Print/export

[Create a book](#)

[Download as PDF](#)

[Printable version](#)

Languages

[Deutsch](#)

[Français](#)

[Italiano](#)

[Nederlands](#)

[Polski](#)

[Русский](#)

[Edit links](#)

Article [Talk](#)

[Read](#)

[Edit](#)

[More](#) ▾



# Clipping (computer graphics)

From Wikipedia, the free encyclopedia



This article **possibly contains original research**. Please [improve it](#) by [verifying](#) the claims made and adding [inline citations](#). Statements consisting only of original research should be removed. *(August 2015)*



This article **needs additional citations for verification**. Please help [improve this article](#) by [adding citations to reliable sources](#). Unsourced material may be challenged and removed. *(August 2015)*

**Clipping**, in the context of [computer graphics](#), is a method to selectively enable or disable [rendering operations](#) within a defined [region of interest](#). Mathematically, clipping can be described using the terminology of [constructive geometry](#). A rendering algorithm only draws [pixels](#) in the [intersection](#) between the clip region and the scene model. Lines and surfaces outside the view volume are removed.<sup>[1]</sup>

Clip regions are commonly specified to improve render performance. A well-chosen clip allows the renderer to save time and energy by skipping calculations related to pixels that the user cannot see. Pixels that will be drawn are said to be within the clip region. Pixels that will not be drawn are outside the clip region. More informally, pixels that will not be drawn are said to be "clipped."

## Contents [\[hide\]](#)

- Clipping in 2D graphics
- Clipping in 3D graphics
  - Occlusion clipping (Z- or depth clipping)
- Importance of clipping in video games
- Algorithms
- See also
- Further reading
- References

## Clipping in 2D graphics [\[edit\]](#)

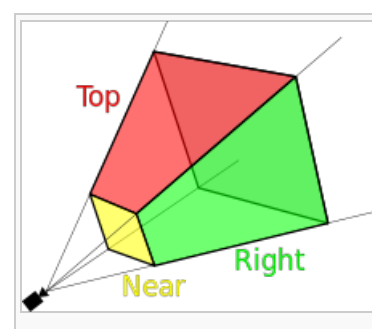
In two-dimensional graphics, a clip region may be defined so that pixels are only drawn within the boundaries of a [window](#) or frame. Clip regions can also be used to selectively control pixel rendering for aesthetic or artistic purposes. In many implementations, the final clip region is the composite (or intersection) of one or more application-defined shapes, as well as any system hardware constraints

In one example application, consider an image editing program. A user application may render the image into a viewport. As the user zooms and scrolls to view a smaller portion of the image, the application can set a clip boundary so that pixels outside the viewport are not rendered. In addition, [GUI widgets](#), overlays, and other [windows or frames](#) may obscure some pixels from the original image. In this sense, the clip region is the composite of the application-defined "user clip" and the "device clip" enforced by the system's software and hardware implementation.<sup>[2]</sup> Application software can take advantage of this clip information to save computation time, energy, and memory, avoiding work related to pixels that aren't visible.


## Clipping in 3D graphics [\[edit\]](#)

In three-dimensional graphics, the terminology of clipping can be used to describe many related features. Typically, "clipping" refers to operations in the plane that work with rectangular shapes, and "culling" refers to more general methods to selectively process scene model elements. This terminology is not rigid, and exact usage varies among many sources.

Scene model elements include geometric primitives: points or vertices; line segments or edges; polygons or faces; and more abstract model objects such as curves, splines, surfaces, and even text. In complicated



scene models, individual elements may be selectively disabled (clipped) for reasons including visibility within the viewport ([frustum culling](#)); orientation ([backface culling](#)), obscuration by other scene or model elements ([occlusion culling](#), depth- or "z" clipping). Sophisticated algorithms exist to efficiently detect and perform such clipping. Many optimized clipping methods rely on specific hardware acceleration logic provided by a [graphics processing unit](#) (GPU).

 A view frustum, with near- and far-clip planes. Only the shaded volume is rendered.

The concept of clipping can be extended to higher dimensionality using methods of abstract [algebraic geometry](#).

### Occlusion clipping (Z- or depth clipping) [\[edit\]](#)

*Main articles:* [depth buffer](#) and [occlusion culling](#)

In 3D computer graphics, "Z" often refers to the depth axis in the system of coordinates centered at the viewport origin: "Z" is used interchangeably with "depth", and conceptually corresponds to the distance "into the virtual screen." In this coordinate system, "X" and "Y" therefore refer to a conventional [cartesian coordinate](#) system laid out on the user's screen or [viewport](#). This viewport is defined by the geometry of the [viewing frustum](#), and parameterizes the [field of view](#).

Z-clipping, or depth clipping, refers to techniques that selectively render certain scene objects based on their depth relative to the screen. Most graphics toolkits allow the programmer to specify a "near" and "far" clip depth, and only portions of objects between those two planes are displayed. A creative application programmer can use this method to render visualizations of the interior of a 3D object in the scene. For example, a [medical imaging](#) application could use this technique to render the organs inside a human body. A video game programmer can use clipping information to accelerate game logic.<sup>[3]</sup> For example, a tall wall or building that occludes other game entities can save GPU time that would otherwise be spent transforming and texturing items in the rear areas of the scene; and a tightly integrated software program can use this same information to save CPU time by optimizing out game logic for objects that aren't seen by the player.<sup>[3]</sup>

## Importance of clipping in video games [\[edit\]](#)



This section **does not cite any references or sources**. Please help improve this section by [adding citations to reliable sources](#). Unsourced material may be challenged and [removed](#). *(August 2015)*

Good clipping strategy is important in the development of [video games](#) in order to maximize the game's [frame rate](#) and visual quality. Despite [GPU chips](#) that are faster every year, it remains computationally expensive to [transform](#), [texture](#), and [shade](#) polygons, especially with the multiple texture and shading passes common today. Hence, [game developers](#) must live within a certain "budget" of polygons that can be drawn each video frame.

To maximize the game's visual quality, developers prefer to let aesthetic choices, rather than hardware limitation, dictate the polygon budget. Optimizations that save performance therefore or take advantage of graphics pipeline acceleration improve the player's experience.

Clipping [optimization](#) can speed up the rendering of the current scene, economizing the use of render time and memory within the hardware's capability. Programmers often devise clever [heuristics](#) to speed up the clipper, as it is sometimes computationally prohibitive to use line casting or [ray tracing](#) to determine with 100% accuracy which polygons are not within the camera's [field of view](#). Spatially aware data structures, such as [octrees](#) and [R\\* trees](#) can be used to partition scenes into rendered and non-rendered areas.

Occlusion optimizations based on viewpoint geometry may introduce artifacts if the scene contains reflective surfaces. A common technique, [reflection mapping](#), can optionally use existing occlusion estimates from the viewpoint of the main view frustum; or, if performance allows, a new occlusion map can be computed from a separate camera position.

For historical reasons, some video games used [collision detection](#) optimizations with identical logic and hardware acceleration as the occlusion test. The terminology "clip" (and its antonym "[no clipping](#)") has sometimes been used to refer to collision detection. "Clip through" may refer to the situation in which part of a model passes through part of another in an unnatural manner, like a leg passing through a cape when running.

## Algorithms [\[edit\]](#)

- **Line clipping algorithms:**
  - [Cohen–Sutherland](#)
  - [Liang–Barsky](#)

- [Fast-clipping](#)
- [Cyrus–Beck](#)
- [Nicholl–Lee–Nicholl](#)
- [Skala](#)
- [O\(lg N\) Algorithm](#)
- **Polygon clipping algorithms:**
  - [Greiner-Hormann](#)
  - [Sutherland–Hodgman](#)
  - [Weiler–Atherton](#)
  - [Vatti](#)
- [Rendering Methodologies](#)
  - [Painter's algorithm](#)

## See also [edit]

- [Boolean operations on polygons](#)
- [Bounding volume](#)
- [Hidden surface determination](#)
- [Pruning \(decision trees\)](#)
- [Visibility \(geometry\)](#)

## Further reading [edit]

- [GPU Gems: Efficient Occlusion Culling](#) <sup>[3]</sup>
- [Clipping in Java AWT](#): `java.awt.Graphics.clipRect` [JavaDoc](#) <sup>[2]</sup>
- [Clipping in UIKit for iOS \(2D\)](#): `UIRectClip` <sup>[1]</sup>
- [Clipping in SceneKit for iOS \(3D\)](#): `SCNCamera` (*Adjusting Camera Perspective*) <sup>[1]</sup>
- [Clipping in OpenGL](#): [OpenGL Technical FAQs: Clipping, Culling, and Visibility Testing](#) <sup>[4]</sup>

## References [edit]

- ↑ Bertoline, Gary; Wiebe, Eric (2002). *Fundamentals of Graphics Communication* <sup>[1]</sup> (3rd ed.). McGraw-Hill. p. G-3. ISBN 0-07-232209-8. Retrieved 2015-01-04.
- ↑ <sup>a</sup> <sup>b</sup> "[java.awt.Graphics.clipRect](#)" <sup>[1]</sup>. Oracle. 2014.
- ↑ <sup>a</sup> <sup>a</sup> <sup>b</sup> <sup>c</sup> Sekulic, Dean (2004). "Efficient Occlusion Culling" <sup>[1]</sup>. *GPU Gems* <sup>[1]</sup>. Pearson. Retrieved 2015-01-02.
- ↑ <sup>a</sup> Paul Martz (2001). "Clipping, Culling, and Visibility Testing" <sup>[1]</sup>. *OpenGL.org*. Retrieved 2015-01-02.

Categories:  Clipping (computer graphics) |  Computer graphics

This page was last modified on 16 August 2015, at 21:54.

Text is available under the [Creative Commons Attribution-ShareAlike License](#); additional terms may apply. By using this site, you agree to the [Terms of Use](#) and [Privacy Policy](#). Wikipedia® is a registered trademark of the [Wikimedia Foundation, Inc.](#), a non-profit organization.

[Privacy policy](#) [About Wikipedia](#) [Disclaimers](#) [Contact Wikipedia](#) [Developers](#) [Mobile view](#)

