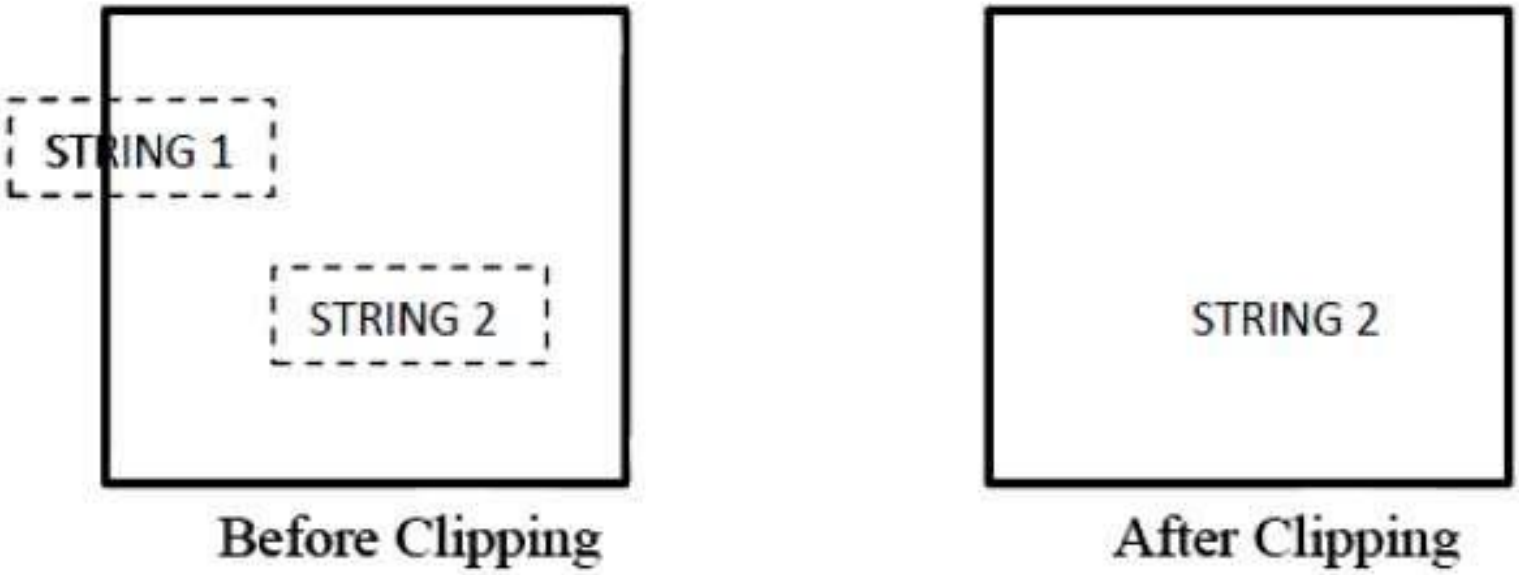
TEXT CLIPPING

- There are several techniques that can be used to provide text clipping in a graphics package. The clipping technique used will depend on the methods used to generate characters and the requirements of a particular application.
- The simplest method for processing character strings relative to a window boundary is to use the **all-or-none string-clipping strategy shown in next Figure.**

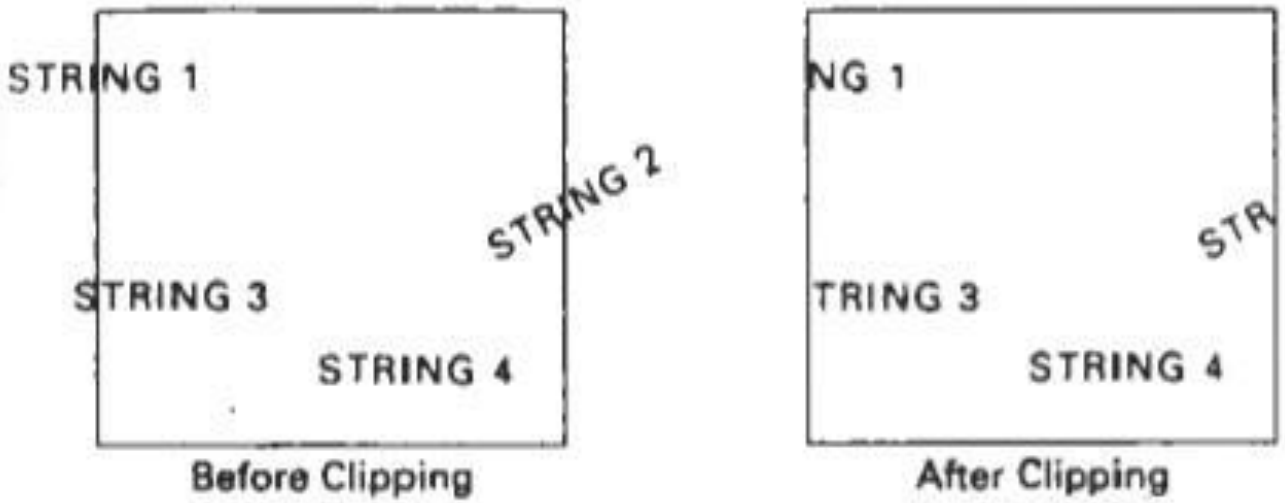
Text clipping using a bounding rectangle about the entire string



- If all of the string is inside a clip window, we keep it.
- Otherwise, the string is discarded.
- This procedure is implemented by considering a bounding rectangle around the text pattern.
- The boundary positions of the rectangle are then compared to the window boundaries, and the string is rejected if there is any overlap.
- This method produces the fastest text clipping.

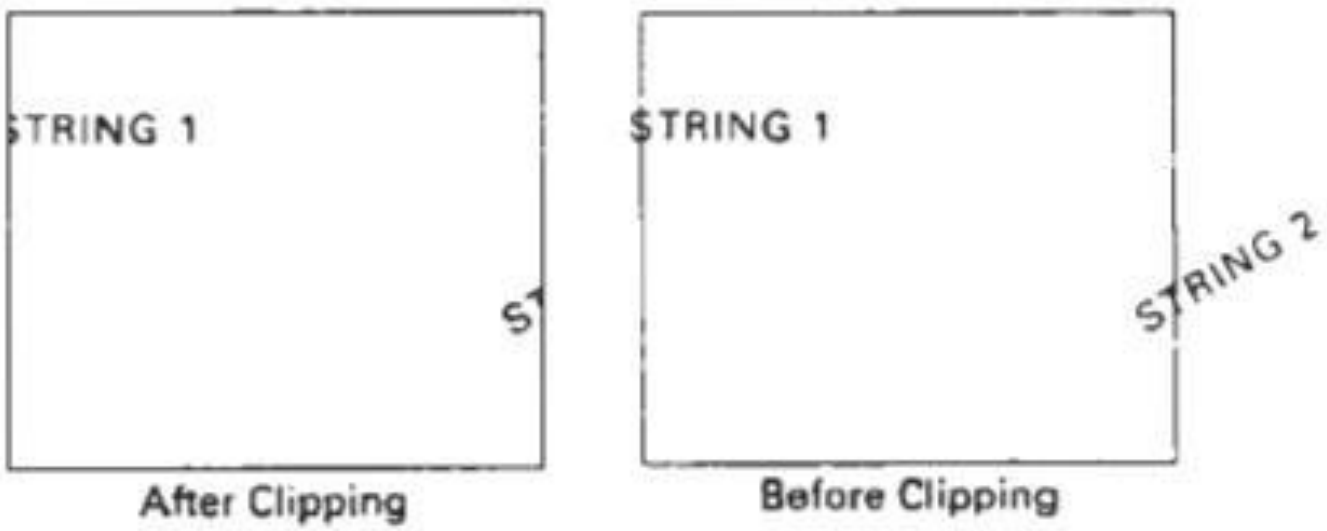
- An alternative to rejecting an entire character string that overlaps a window boundary is to use the all-or-none **chararacter-clipping strategy.**
- **Here we discard only** those characters that **are not completely inside the window .**
- In this case, the boundary limits of individual characters are compared to the window.
- **Any character that either overlaps or is outside a window boundary is clipped**

Text clipping using a bounding rectangle about individual characters



- **A final method for handling text clipping is to clip the components of individual** characters.
- We now treat characters in much the same way that we treated lines.
- If an individual character overlaps a clip window boundary, we clip off the parts of the character that are outside the window.

Text clipping performed on the individual characters



- Outline character fonts formed with line segments can be processed in this way using a line clipping algorithm.
- Characters defined with bit maps would be clipped by comparing the relative position of the individual pixels in the character grid patterns to the clipping boundaries.

Thanks