# Adobe Postscript

**Evin Aydin**

**Specialisation Seminar**

Supervisor:
Assoz. Prof. Dr. Cezary Kaliszyk
Department of Computer Science
Universität Innsbruck

Innsbruck, June 30, 2023

**Abstract**

This comprehensive report provides a deep insight into the programming language PostScript. It covers not only basic concepts and ideas but also extensively explains the various functionalities and the original purpose for which PostScript was developed. The main focus is on examining PostScript as an extremely powerful programming language. Additionally, the limitations of PostScript are discussed and analyzed. Furthermore, a comparison is made between PostScript and the programming language *TEX* to highlight the differences and similarities.

## 1 Introduction

PostScript, developed by Adobe Systems in the 1980s, is a programming language designed as a page description language for printers and printing machines. It has revolutionized the creation of complex documents by integrating text, graphics, and images that can be output on any PostScript-compatible printer or device. What sets PostScript apart is its ability to describe vector graphics. Instead of creating graphics from individual pixels, PostScript uses mathematical descriptions of lines, curves, and shapes. This enables scalable graphics that can be printed in high quality without losing resolution. PostScript files contain a variety of instructions and commands that are interpreted by an integrated PostScript interpreter. This interpreter is built into a range of devices, including printers and image processing devices, as well as some software applications. By reading and executing these instructions, the interpreter generates the desired printing output or corresponding image output.

Overall, PostScript is a versatile programming language specifically designed for professional use in the printing and graphics industry. It continues to be widely used in various fields, including prepress, digital image processing, and page layout software.

## 2 History of Postscript

The founding story of PostScript is closely intertwined with the visionary minds of John Warnock and Charles Geschke, who contributed significantly to the development of this groundbreaking programming language during their time at Xerox PARC[1] in the 1970s and 1980s. Their groundbreaking insights into the need for a universal page description language led to the establishment of Adobe Systems in 1982.

The fundamental problem that was recognized at the time was the challenge of consistently rendering fonts and graphics across different printers with varying resolutions. Previous approaches based on bitmaps were unable to provide the desired quality and flexibility.

[1]Official Website of PARC:`https://www.parc.com`, visited 01.06.2023

To address this challenge, an innovative method was devised: instead of relying on bitmaps, fonts and graphics were described using mathematical curves. Precise adjustments of these curves allowed them to align perfectly with the pixel grids of the respective printers. This enabled high-quality and scalable rendering of fonts and graphics on different devices without sacrificing resolution or quality.

This solution had a revolutionary impact on the computer industry. Customers, printer manufacturers, and experts immediately recognized the immense potential of PostScript for font rendering and the creation of high-quality graphics. Through partnerships with renowned companies such as Apple[2], Linotype[3], DEC, Wang, QMS, and Compugraphic[4], PostScript[5] became an industry standard.

The successful resolution of the font and graphics challenge marked a milestone for Adobe. PostScript, along with its evolution into PDF, played a crucial role in the transformation of the printing and publishing industry worldwide. The precise rendering of fonts and the ability to create high-quality graphics on different devices paved the way for the success and recognition of Adobe.[1]

## 3 The Postscript Language

PostScript is a stack-based programming language that utilizes postfix notation, also known as reverse Polish notation. This notation allows for efficient and unambiguous representation of mathematical expressions. PostScript provides powerful features such as loops, recursion, and conditional branching, enabling the solution of complex tasks.[2, Page 23-24] With these capabilities, PostScript is considered a Turing-complete language[3], capable of solving any computable task. Its versatility and power find application in areas such as graphic design, special effects generation, and automated document processing.

### 3.1 Generating PostScript files

There are several ways to create PostScript files. One option is to learn the PostScript language and manually write the source code. However, there are also automatic methods where system drivers or application programs generate PS files. Additionally, PS converters or filters can be used to automatically convert other file formats into PostScript. The result is device-independent PS files that can be printed on different devices.[4]

[2]Official Website of Apple:`https://www.apple.com`, visited 01.06.2023
[3]Official Website of Linotype:`https://www.linotype.com/de`, visited 01.06.2023
[4]Official Website of Compugraphics:`https://www.compugraphics-photomasks.com/de/`, visited 01.06.2023
[5]Official Website of Adobe:`https://www.adobe.com`, visited 01.06.2023

## 3.2 Interpreter

There are multiple PostScript interpreters because, PostScript is designed to be an interpreted language, allowing flexibility and adaptability to support various environments and devices. The PostScript interpreter is typically a component of software or hardware that processes PostScript files. The PostScript interpreter controls the actions of the output device based on the instructions provided in a PostScript program. There are three types of interaction: conventional printing mode, integrated display model, and interactive programming model, which can be seen in the table below 3. The interpreter can be used for both printers and displays. In an interactive session, a programmer can directly interact with the interpreter. The page description should adhere to structuring conventions to facilitate document interchange and management.[2, Page 15]

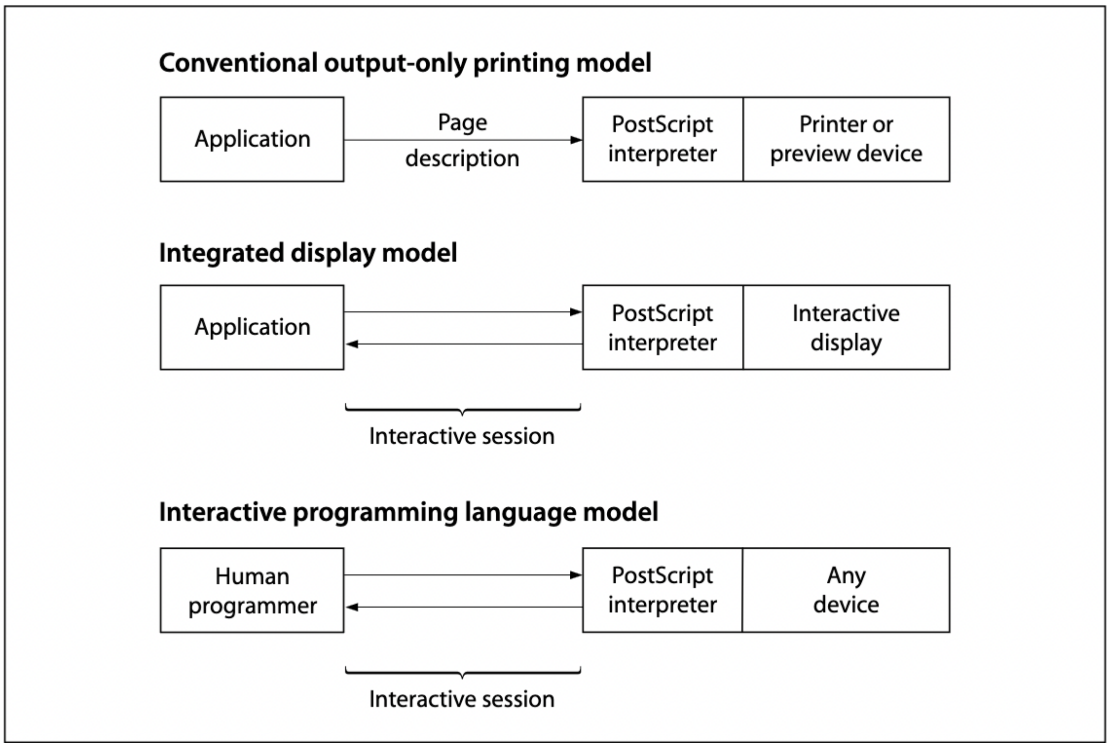
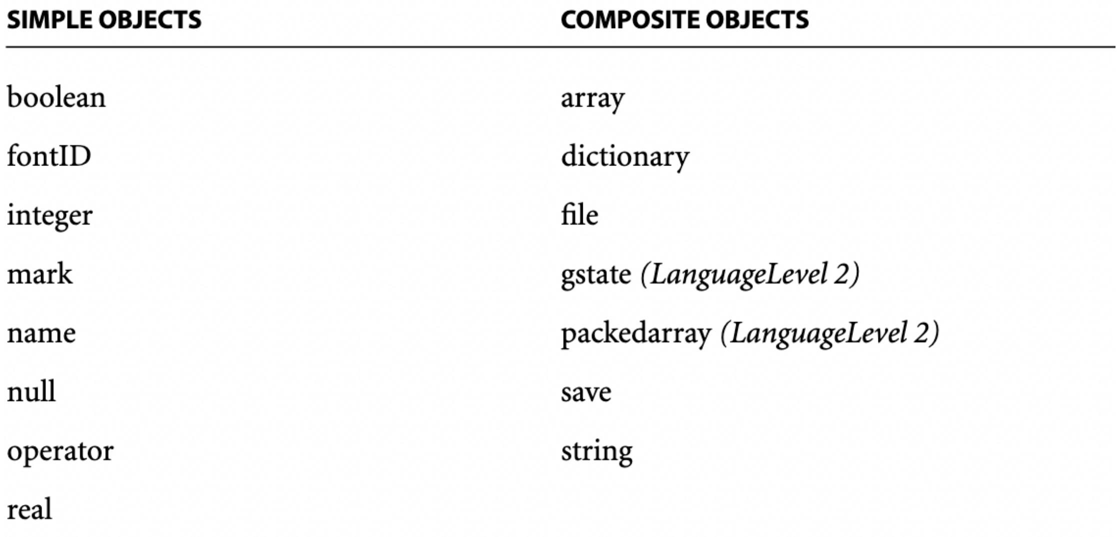


Figure 1: Interaction between PostScript interpreter and application [2, Page 16]

## 3.3 Basicelements

In PostScript, all data, including procedures, exists in the form of objects, which are created, manipulated, and used by operators. Each object has a type, attributes, and a value, where the type is a property of the object itself. The PostScript language supports various object types, as listed in Table below and additional types can be introduced through language extensions.



| SIMPLE OBJECTS | COMPOSITE OBJECTS |
|---|---|
| boolean | array |
| fontID | dictionary |
| integer | file |
| mark | gstate *(LanguageLevel 2)* |
| name | packedarray *(LanguageLevel 2)* |
| null | save |
| operator | string |
| real | |

Figure 2: Types of objects [2, Page 34]

In PostScript, there are two types of data: simple data types and composite data types. sSimple data types in PostScript consist of value, data type, and associated attributes held together as a package. When copying such an object, all these elements are replicated. However, for composite objects, the values are not copied but rather the copy refers to the original values, similar to pointers in C. [2, 34-35]

### 3.3.1 The simple data types

#### Number Objects

- Signed integers, such as 123 98 43445 0 +17
- Real numbers, such as .002 34.5 3.62 123.6e10 1.0E5 1E6 1. 0.0
- Radix numbers, such as 81777 16FFFE 21000

An integer consists of an optional sign followed by one or more decimal digits. It is interpreted as a signed decimal integer and converted to an integer object. If it exceeds the implementation limit, it is converted to a real object. A real number includes a sign, decimal digits, and optional exponent. It is interpreted as a floating-point number and converted to a real object. If it exceeds the implementation limit, an error occurs. Radix numbers have a base and are interpreted as unsigned integers with the corresponding binary representation. [2, Page 28]

**Boolean**
PostScript uses boolean objects (true and false) for conditions and logic. They are generated by comparison and logical operators and used as operands for "if" and "ifelse" control operators. [2, Page 38]

**Name Objects**
In PostScript, a name is a data type that represents unique identifiers for variables, procedures, and other entities. It is defined by a sequence of characters and allows for efficient equality checking without the need to compare the actual characters themselves. [2, Page 40]

**Mark Objects**
A marking object (mark) indicates a position in the operand stack and is used in certain stack and array operations. [2, Page 44]

**Null objects**
The PostScript interpreter utilizes null objects to occupy empty or uninitialized positions within composite objects upon their creation. [2, Page 44]

**Operator object**
An operator object in PostScript represents a predefined action of the language that is associated with names in the systemdict data structure. Standard operators are defined in systemdict and can be modified by PostScript programs. An example of an operator in PostScript is the "add" operator, which adds two numbers. [2, Page 42-43]

**FontId Objects**
FontID objects in PostScript accurately specify and utilize fonts, ensuring consistent rendering and abstraction of font implementation details. [2, Page 45]

### 3.3.2 The composite data types

**Strings**
String objects can be quoted using three conventions:

- Enclosed in parentheses ( ) as literal text
- Enclosed in angle brackets < > as hexadecimal data
- Enclosed in <~ ~> as ASCII base-85 data (LanguageLevel 2). [2, Page 29]

**Array Objects**
A PostScript array is a collection of objects that can be accessed using a numerical index. Arrays can contain different elements and are indexed starting from 0. When copying an array, the old and new objects share the same value. [2, Page 38-39]

**Packedarray Objects**

A packed array is a more compact representation of an ordinary array, primarily used as a procedure. It occupies less memory and behaves similarly to a regular array. During execution, the two types of arrays are indistinguishable unless they are treated as data. [2, Page 39]

For example:[2 3 add]

**Dictionary Objects**

A PostScript dictionary is a table of associative pairs of PostScript objects. Each entry consists of a key and an associated value. Dictionaries allow for insertion, searching, and retrieval of values based on the keys. Keys are usually name objects but can also be other PostScript objects, except for null. Dictionaries have a maximum number of entries and can store a specific number of elements. There are various methods to access dictionaries, including specific operators and the current dictionary. When copying a dictionary, the contents are not duplicated but shared. [2, Page 41]

**File objects**

A PostScript file is a readable or writable stream of characters between the interpreter and the environment. File objects allow opening, reading, writing, and processing characters. When copied, they share the underlying file. [2, Page 43-44]

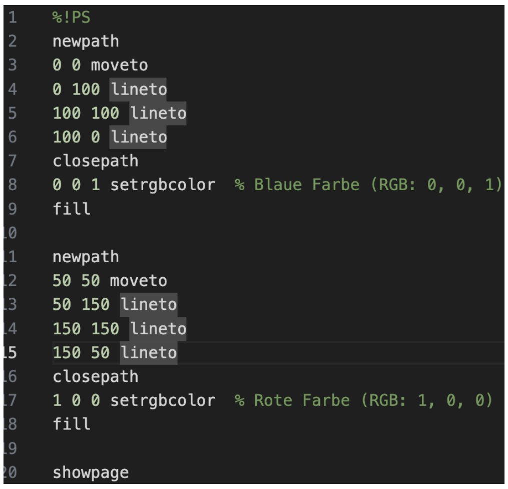
**Save objects**

Save objects are used to store and restore the memory state of the PostScript interpreter. The save operator creates a save object that holds a copy of the current memory state. The restore operator is used to restore the previously saved state. This allows for storing multiple states and switching between them, which is particularly useful for undoing changes or performing alternative computations. [2, Page 44-45]

## 3.4 Stack

The PostScript interpreter maintains five stacks: operand, dictionary, execution, graphics state, and clipping path stacks.

- The operand stack stores objects used as operands and operator results.
- The dictionary stack holds dictionaries for name searches.
- The execution stack manages intermediate execution stages.
- The graphics state stack controls graphics parameters.
- The clipping path stack handles the current clipping path. [2, Page 45-46]


```
%!PS
newpath
0 0 moveto
0 100 lineto
100 100 lineto
100 0 lineto
closepath
0 0 1 setrgbcolor  % Blaue Farbe (RGB: 0, 0, 1)
fill

newpath
50 50 moveto
50 150 lineto
150 150 lineto
150 50 lineto
closepath
1 0 0 setrgbcolor  % Rote Farbe (RGB: 1, 0, 0)
fill

showpage
```

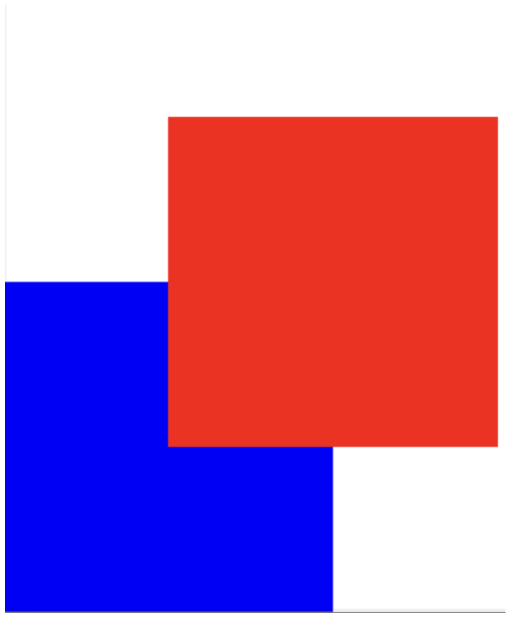


Figure 3: Codeexample with output

## 4 Distinguished Features of Postscript

### 4.1 Graphs

The combination of text and graphics can significantly enhance the presentation of information. While traditional books and catalogs can fulfill this function, computer-based databases offer a practical alternative. However, they often face limitations as many database management systems do not support the integration of graphical elements or the visual representation of query results. One solution to this problem is the programming language PostScript. PostScript enables the creation and integration of high-quality graphics into databases, which can greatly improve the user experience. This language is based on a model where ink is selectively applied to specific areas to generate images. It is device-independent and provides high flexibility in designing graphical elements. [5] Therefore, PostScript graphics operators are a versatile tool for controlling the appearance of pages on raster output devices. They allow for the manipulation of the graphics state, including the transformation of coordinate systems and matrices to adapt to the output device.This can be seen in Figure 4. The path construction operators define shapes and line trajectories, while the painting operators enable the drawing and filling of graphical elements such as lines, filled areas, and images. The glyph and font operators provide the selection and rendering of character glyphs from fonts.In PostScript, Glyphs refer to the individual graphical representations of characters or symbols within a font. The device setup operators establish the connection between the raster memory and the physical output device, and the output operators transmit the fully described page to the output device. With this comprehensive set of operators, PostScript offers a flexible and powerful solution for creating and displaying graphic elements on various output devices. [2, Page 175-192]

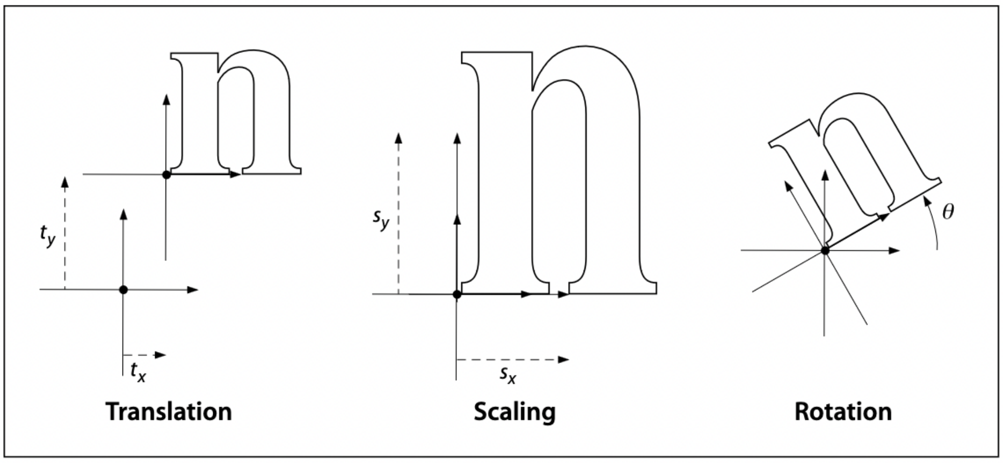


Figure 4: Coordinate Transformations and their effects [2, Page 188]

## 4.2 Fonts

PostScript provides advanced font support. Fonts in PostScript are based on mathematical curve descriptions and are stored as vectors. This allows for precise scaling, rotation, and transformation without any loss of quality. Vector-based fonts offer high resolution and a consistent appearance across different media and sizes. Complex font effects and customizations can be implemented using mathematical formulas to meet individual requirements. [2, Page 315-327] There is a wide selection of fonts available, both preinstalled and downloadable, in various styles and qualities. PostScript enables the display of grayscale and colors, as well as seamless integration of text and graphics. [6]

One notable feature of PostScript is the encryption of fonts. In PostScript font files, the font data is stored in an encrypted format, making it unreadable for users. While the font file contains some readable information at the beginning, such as the font name and character set, the actual outline descriptions of the characters appear as unintelligible gibberish. This encryption was developed by Adobe to ensure the protection of font data. To use the font, the first step is to reverse the so-called "exec encryption." This process decrypts the font data, revealing the CharStrings dictionary that contains the descriptions of individual characters. Then, the CharStrings encryption needs to be resolved, as the data is protected by a second encryption step. Only after these steps are completed, the actual instructions for shaping the letterforms become available. [7, Page 119-144]

## 4.3 Extensions of Postscript

### 4.3.1 EPS

Encapsulated PostScript (EPS) was developed to facilitate the exchange of graphics and other objects between different software applications and operating systems. It is based on PostScript and aims to be independent of the output technology used. Therefore, EPS files have a different use case compared to PS files. Typically, finalized graphics and objects are saved in EPS files, which are then inserted into layouts or printed, often with a size adjustment. [6] An EPS file contains several components. First, it includes the actual PostScript instructions that describe how the graphics should be displayed. These instructions are interpreted by a PostScript interpreter and rendered on a PostScript-compatible printer or screen.

Second, the EPS file contains a preview version of the graphics, which is used for displaying the graphics on screens that do not have a PostScript interpreter. This preview is usually a low-resolution raster image that provides a rough representation of the graphics.

Lastly, the EPS file also contains information about the graphics itself, such as dimensions, colors used, fonts, and other metadata. Desktop publishing programs and other applications utilize this information to properly insert and display the graphics in a document.

EPS files are supported by a variety of graphic design, desktop publishing, and word processing programs. Graphic design programs can export graphics as EPS files, while desktop publishing or word processing programs can import EPS files to insert them into documents.

It's important to note that EPS and PostScript are closely related but not identical. EPS is a specialized variant of the PostScript format specifically designed for graphic exchange.

EPS files can be created in various ways. Graphic design programs typically provide an export function to save graphics in the EPS format. There are also specialized converters available that can convert other graphic formats into EPS. Additionally, printer drivers can generate EPS files by redirecting the output to this format.

When embedding an EPS graphic into a document, size and position can be adjusted even if the importing program does not have its own PostScript interpreter. This is achieved through specific instructions that scale, move, or crop the graphic. When printing, these instructions are transmitted along with the PostScript data to the printer to accurately render the graphics. [7, Page 27-39]

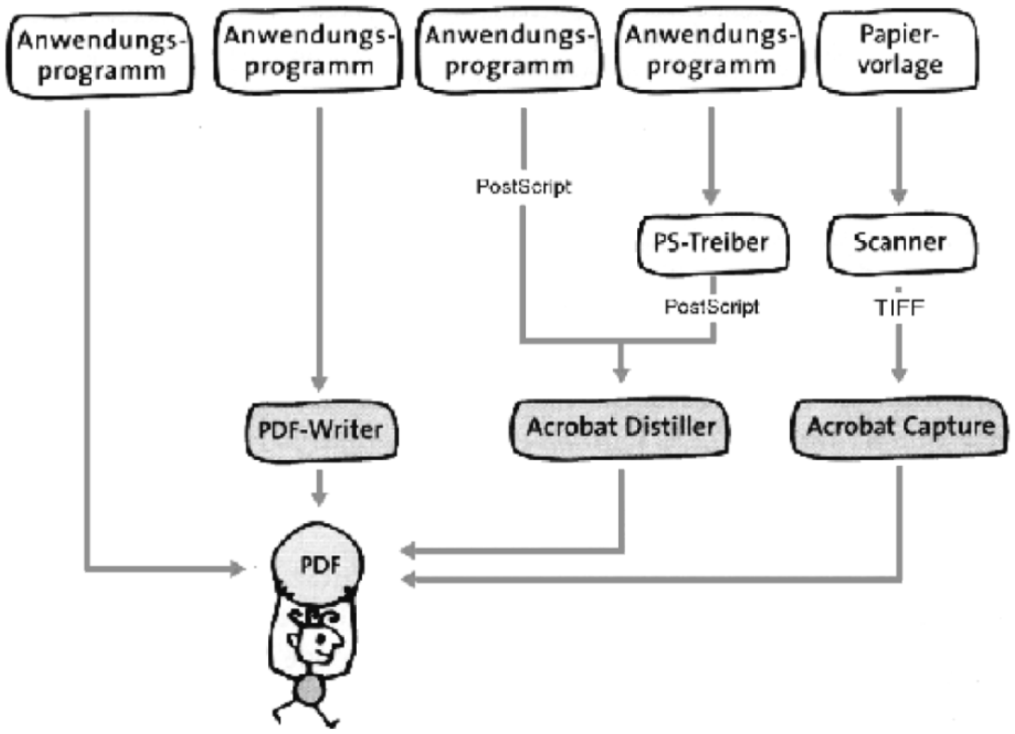


Figure 5: Creating a PDF file [8]

### 4.3.2 PDF

PDF (Portable Document Format) was developed by Adobe as an evolution and complement to PostScript, addressing the limitations of the latter and providing a versatile file format for document exchange. While PostScript focused primarily on print output, PDF extended its functionality to enable viewing and interaction with documents on screens.

One significant advantage of PDF over PostScript is its platform independence. PDF files can be viewed consistently across different operating systems and devices, ensuring the preservation of document formatting and layout. This portability allows for seamless document sharing and collaboration, regardless of the hardware or software used.To view, edit, and convert PostScript files to PDF, Adobe Acrobat is commonly used. Acrobat is a comprehensive software package that includes various tools for working with PDFs. It encompasses Acrobat Reader, which is a widely used application for viewing PDF files, providing a user-friendly interface for navigating and interacting with the document's contents. This can be seen in the Figure 5.

PDF offers enhanced features beyond what PostScript provided. It supports interactive elements like hyperlinks, forms, multimedia content, and annotations, facilitating a richer user experience. Additionally, PDF includes advanced security features such as password protection, encryption, and digital signatures, ensuring the confidentiality and integrity of sensitive information.

Compared to PostScript, PDF also offers superior compression capabilities, allowing for efficient storage and transmission of documents without compromising their quality. The ability to compress PDF files helps optimize file sizes, making them more manageable and easier to distribute.

Moreover, PDF files can be opened directly by applications like Adobe Acrobat Reader without the need for a PostScript interpreter. This standalone nature of PDF simplifies document sharing and viewing, as it eliminates the requirement for an additional software component to interpret the graphics and display them accurately.

PDF's versatility extends beyond print-centric applications. It has become a widely adopted standard in various industries, including publishing, archiving, digital workflows, and online content distribution. With its support for structured data, indexing, and searchability, PDF enables efficient information retrieval and extraction from documents.

In summary, PDF emerged as an advanced file format that builds upon PostScript's foundations while addressing its limitations. It offers platform independence, interactive features, enhanced security, efficient compression, and broad industry adoption, making it an essential tool for digital document management and sharing. [9]

## 4.4 Scalable Vector Graphics

SVG (Scalable Vector Graphics) is widely used as the dominant markup language for web visualizations due to its openness, accessibility, and easy customization with CSS. However, as datasets and the corresponding number of DOM nodes increase, SVG-based visualizations can suffer from slow performance. To address this limitation, the research explores decoupling the SVG specification from browser-based rendering. They propose Scalable Scalable Vector Graphics (SSVG), a JavaScript library that enhances SVG performance by utilizing a virtual DOM, offloading rendering to worker threads, and enabling element-level interactivity. SSVG improves rendering speed, reduces DOM overhead, and achieves parallel rendering, making it a promising solution for high-performance web-based information visualization. [10]

# 5 Limitations of Postscript

### 5.0.1 Transparency Problem

In graphic design, the process of extracting objects from their backgrounds, known as "clipping," is a common task. This is done by creating a clipping path that traces the outline of the object to be isolated, making the desired area visible while hiding the rest.

However, PostScript faces challenges when dealing with complex clipping paths that can contain a large number of path segments. This can overload the PostScript interpreter and lead to errors. As an alternative, image masks are used, which contain information for each individual pixel regarding whether it should be displayed or hidden.

There are different types of image masks, such as explicit image masks that include an additional channel for masking information. However, transparency in PostScript is limited to 1 bit, which means that true transparency is not possible. Instead, only the showing or hiding of image portions is achieved. In order to achieve such effects, the software generating the PostScript must manually calculate the mixture of foreground and background and simulate the desired effect. However, this often results in the loss of the original structure of the objects, as overlapping objects need to be divided into many small parts (flattening process). [7, Page 13-15]

### 5.0.2 Implementation Problem

The PostScript language itself does not impose any inherent limitations on the size or quantity of elements such as numbers, arrays, stacks, etc. However, specific PostScript interpreters running on particular hardware and operating environments have certain limitations that cannot be exceeded. If a PostScript program surpasses these limits, it triggers a "limitcheck" error (or a "VMerror" if virtual memory resources are exhausted). [2, Page 737-743]

PostScript implementations have two types of limits architectural limits and memory limits.

- Architectural limits refer to hardware-imposed constraints of the PostScript interpreter. For example, integers are typically limited to 32 bits, which restricts the range of values. There are also constraints set by the software design, such as a limit of 65,535 elements in arrays or strings.
- Memory limits pertain to the amount of available memory for the PostScript interpreter. These limits determine the number of memory-intensive objects the interpreter can simultaneously hold. Memory management varies depending on the product. [2, Page 738]

# 6 Postscriped compared to LaTeX

LaTe$X$ is a typesetting and writing program for various types of documents. It provides high stability, excellent representation of mathematical formulas, and professional layout. It is commonly used in scientific and technical fields. Many international journals require the use of LaTe$X$ for manuscripts. Workshops cover a range of topics, from basic to advanced, to learn how to use LaTe$X$ effectively. However during the compilation of a LaTe$X$ document, the resulting output format can include various formats, including PostScript. [11] On the other hand, PostScript is a page description language specifically designed for printing documents. It is a programming language that allows the description of graphics and text elements to be positioned on a physical page. [2] Indeed, both PostScript and LaTeX are Turing-complete languages, which means that they are theoretically capable of executing any computable function. To better illustrate the syntax differences, we will conduct a detailed analysis of the syntax of both languages.

## 6.1 Postscript Syntax

- PostScript uses a stack-based syntax, where commands and operands are placed on a stack in reverse order.
- PostScript employs reverse Polish notation, where operators are placed after their operands.
- The syntax encompasses functions for path construction, text output, and execution of arithmetic operations.
- The command structure consists of a command followed by its arguments, separated by spaces or line breaks.
- Conditions and loops enable the execution of actions based on specific criteria. [2]

## 6.2 LaTeX Syntax

- LaTeX utilizes a markup syntax, employing commands and environments for structuring and formatting text.
- The syntax encompasses defining the document structure, formatting text, incorporating figures and tables, and representing mathematical formulas.
- Citations and references can be generated using special commands and bibliographic databases
- Commands in LaTeX are introduced by a backslash , followed by the command name. [12]

## 6.3 Using Postscript in LaTeX

The special command enables the direct incorporation of PostScript instructions into a LATE$X$document and facilitates seamless interaction with TE$X$. By utilizing this command, a PostScript graphic can be precisely positioned within the document.

To embed a PostScript graphic, the TE$X$ document includes the special command followed by parameters like the filename of the PostScript file, along with the desired height and width of the graphic. This command is placed at the specific location in the document where the graphic should be displayed.

Upon processing the TE$X$ document, the Dvi/PS processor interprets the special command and executes the corresponding instructions. The graphic is appropriately scaled based on the specified dimensions and accurately placed on the page. [13]

Furthermore, it is also feasible to incorporate PostScript fonts in documents. This involves utilizing special commands and packages in LATE$X$ that allow for font selection and integration according to the desired design. [8]

- Helvet
  The package provides the Helvetica font, which is slightly larger in size compared to other fonts.
- Mathpazo
  The package changes the default roman font to Adobe Palatino and utilizes virtual "mathpazo" fonts for mathematical formulas.
- Mathptmx The mathptmx package modifies the default roman font family to Times and employs virtual mathptmxfonts for mathematical formulas. When loaded with the [slantedGreek] option, it italicizes uppercase Greek letters in math. Additionally, it introduces new commands such as upGamma, upDelta, etc. for obtaining upright Greek uppercase letters. Notably, the package automatically scales "large" math symbols to fit the base font size, eliminating the need to load the exscale package.
- Mathpple
  Mathpple is a predecessor to Mathpazo, using Greek letters from the Euler fonts and having some flaws. Mathpazo, on the other hand, is an improved version that supports the Palatino SC/OsF fonts.

# 7 Conclusion

The analysis of the programming language PostScript and its significance in the printing and graphic industry reveals that PostScript is a versatile language. With a wide range of functions and operators, it enables complex graphic effects and transformations. The seamless integration of text and graphics facilitates workflow and data transfer between different systems in the printing industry. PostScript is indispensable for generating high-quality, scalable graphics and serves as the foundation for other important formats such as PDF. Despite the emergence of new technologies, PostScript remains relevant and continues to play a crucial role in modern printing and graphic applications. Overall, PostScript is a flexible programming language that is essential for creating sophisticated materials.

## References

[1] John E. Warnock. The origins of postscript. *IEEE Annals of the History of Computing*, 40(3):68–76, 2018.

[2] Adobe Systems Incorporated. *Postscript Language Referencecs third Edition.*

[3] Gene Michael Stover. Printer hacking. 2004.

[4] Hannelore Schmidt and Susanne Dobratz. Postscript und pdf. 1999.

[5] Kelvin Goodson. Postscript for archaeological drawings. *Computer and Quantitative Methods in Archaeology 1988*, pages 335–352, 1988.

[6] Don Lancaster. What is postscript? a tutorial. *ACM SIGFORTH Newsletter*, 3(2):15–19, 1991.

[7] Thomas Merz. *Die PostScript- und PDF-Bibel.* PDFlib GmbH, Berlin, 1996.

[8] Walter Schmidt. Using common postscript fonts with latex. *Bestandteil der Online-Dokumentation von LATEX (seit Juni 2000), Datei psnfss2e. pdf*, 2004.

[9] Electronic File Exchange. Portable document format (pdf)—finally, a universal document exchange technology. *Promoting Excellence in Preparation and Excellence in Practice*, 415:59, 2002.

[10] Michail Schwab, David Saffo, Nicholas Bond, Shash Sinha, Cody Dunne, Jeff Huang, James Tompkin, and Michelle A. Borkin. Scalable scalable vector graphics: Automatic translation of interactive svgs to a multithread vdom for fast rendering. *IEEE Transactions on Visualization and Computer Graphics*, 28(9):3219–3234, 2022.

[11] Yogesh Singh and Shri PR Sheth. Latex.

[12] Tobias Oetiker, Hubert Partl, Irene Hyna, and Elisabeth Schlegl. The not so short introduction to latex. *Typeset LATEX 2ε*, 2001.

[13] Alec Dunn. Using postscript with ll&x.