Visualization and Animations in Mathematics Using PostScript and PDF

Prof. dr Miroslav Ćirić

University of Niš

Faculty of Sciences and Mathematics

Introduction

This lecture is a continuation of the talks given at

- **Workshop on Innovations in Teaching of Computer Science Courses, Niš, October, 2003;**
- **Workshop on Teaching Methods in Computer Science Courses, Skopje, October, 2003;**
- **Symposium on Innovation of Computer Science Curriculum in Higher Education: Retraining of Young teaching staff, Athens, February, 2004;**

- **Seminar for Elementary and Secondary School Teachers in Mathematics and Informatics, January, 2004;**
- **Tempus Workshop on New Methods in Teaching of Mathematics, Novi Sad, October, 2004;**

Also, it is partially based on a course for teachers and teaching assistants on our Department of Mathematics and Informatics, February-March, 2004.

The main goal of this lecture is to demonstrate the most important capabilities of TEX, PostScript (i.e. PSTrics) and PDF, when they work together.

A question that we meet very often in our work is

- **How to prepare a nice looking presentation to be presented to our students or on a scientific conference?**

This problem is especially difficult if the material that has to be presented contains a lot of mathematical symbols and complex mathematical formulas, as well as a lot of graphics.

As we know, the best system for typesetting such material is TEX.

But, what is about graphics?

Remarks on Graphics in TEX

- TEX is s system (program) for typesetting and text processing, intentionally designed for typesetting of material with a lot of mathematical symbols and complex mathematical formulas.
- TEX probably has the best algorithm for formatting paragraphs and building pages from them.
- To a very large extent, TEX was designed for the placement of characters on a page.
- It was implicitly assumed that the characters are alphabetic or mathematical.

- But, TEX is not conceived to be a tool for working with graphics.
- Donald Knuth (TEX's creator) himself noted that

  "If you enjoy fooling around making pictures, instead of typesetting ordinary text, TEX will be a source of endless frustration/amusement for you, because almost anything is possible..."

- However, there are many ways in which graphics may be made part of TEX documents.
- Generally speaking, TEX offers two major facilities for integrating graphics and TEX:
  - using the font interface,
  - involving the `\special` command.

The Font Interface

- This approach is based on use of special fonts to build pictures.
- That can be done in various ways:
  - Through simple font elements (that is, straight line segments, or curves) which can be assembled to give (fairly simple) pictures.
  - Through METAFONT – a programming language used to typeset outline fonts. Here, METAFONT is used to create a single character which is our graphics (or whatever), etc.
- One of the best known examples of this approach is the LaTeX `picture` environment.

- Good sides of this approach are generality and portability.
- Seeing that everything is in TEX, we can also ensure that the relative weights of lines, the font sizes, the symbols, blend in well with the rest of the document.
- The key drawbacks of the special font approach are centered around the limited fonts which are available, both in the slope of lines and their thicknesses, and the limited range of curves.

The Special Command

- Donald Knuth, TEX's creator, provided the `\special` command as a hook for implementing "features" that are not available in the basic language.
- This command does not affect the output page being formatted, but TEX put the material, specified as an argument in the `\special` command, literally at the current point in the `.dvi` file.
- The dvi-driver has to interpret the received information and produce the output accordingly.

- The `\special` command allows us to access special features of a driver program that translates the `.dvi` output of TEX in the language understood by the output device.
- If this driver has mechanisms to include external graphics, then we can import such graphics.
- The price we pay is non-portability, since our source will contain calls to a non-standard interface.
- Authors of different drivers have implemented different conventions so that documents that contain direct calls to `\special` only work with a very restricted setup.
- However, the non-portability problem can often be solved translating the `.dvi` output into PostScript.

Remarks on PostScript

- PostScript is a programming language introduced by Adobe in 1985 and first appeared in the Apple LaserWriter.
- The main purpose of PostScript was to provide a convenient language in which to represent the printed page in a device independent manner.
- This device independence means that the image is described without reference to any specific device features (e.g. printer resolution) so that the same description could be used on any PostScript printer without modification.

- **PostScript is a page description language which describes a complete page at a time, rather than one line at a time, like a line printer.**
- **It is a high-level programming language, which is stack-oriented and uses "reverse Polish" or postfix notation.**
- **PostScript is a flexible language**
  - **it includes looping constructs, procedures, and comparison operators,**
  - **it supports many data types, including reals, Booleans, arrays, and strings,**
  - **and complex objects, such as dictionaries.**

- **Functions that do not exist, but which would be useful for an application, can be defined and then used like other PostScript operators.**
- **Thus, PostScript is not a fixed tool within whose limits an application must be written, but is an environment that can be changed to match the task at hand.**
- **PostScript has a large selection of graphics operators that allow it to precisely describe a desired page.**

- These operators control the placement of three types of graphics objects:
  - **Text** in a wide variety of typefaces can be placed on a page in any position, orientation, and scale.
  - **Geometric figures** can be constructed using PostScript graphics operators. These describe the locations of straight lines and curves of any size, orientation, and width, as well as filled spaces of any size, shape, and color.
  - **Sampled images** of digitized photographs, free-hand sketches, or any other image may be placed on a page in any scale or orientation.
- All graphic objects may be easily rotated, scaled, and clipped to a specified portion of the output page.

Integration of PostScript and TEX

- When a PostScript output device and a dvi-to-ps driver are used to print or display TEX files, TEX and PostScript work together, as a preprocessor and a postprocessor, respectively.
- The role of PostScript may simply be to render TEX's dvi typesetting instructions.
- However, the full power of PostScript can be accessed through `\special`'s and through features, such as font handling, built into the dvi-to-ps driver.

- One can divide the PostScript enhancements to TEX into roughly four categories:
  - The use of PostScript fonts.
  - The inclusion of PostScript graphics files.
  - The coloring of text and rules.
  - Everything else.
- Most TEX-PS users are familiar with the first three categories.
- The most powerful tool for covering the fourth category is the PSTricks macro package.

Remarks on PSTrics

- The PSTricks package started as an implementation of some special features in the Seminar document style/class, which is for making slides with LaTeX2_ε.
- It was created by Timothy Van Zandt (Princeton, USA) in 1994, and further developed by him and Denis Girou (Orsay, France).
- However, it has grown into much more.
- Some of its current features are:
  - Graphics objects (analogous to LaTeX picture commands such as `\line` and `\frame`), including lines, polygons, circles, ellipses, curves, springs and zigzags.

- Other drawing tools, such as a picture environment, various commands for positioning text, and macros for grids and axes.
- Commands for rotating, scaling and tilting text, and 3-D projections.
- Text framing and clipping commands.
- Nodes and node connection and label commands, which are useful for trees, graphs, and commutative diagrams, among other applications.
- Overlays, for making slides.
- Commands for typesetting text along a path.
- Commands for stroking and filling character outlines.
- Plotting macros, etc.

- **A goal of PSTricks was to be compatible with any TEX format and any dvi-to-ps driver.**
- **Compatibility with the various TEX formats is not difficult to achieve, because PSTricks does not deal with page layout, floats or sectioning commands.**
- **However, compatibility with all dvi-to-ps drivers is an unattainable goal because some drivers do not provide the basic `\special` facilities required by PSTricks.**
- **But, all of PSTricks' features work with the most popular driver `dvips`, and most features work with most other drivers.**

- PostScript language is used for performing complex computation within the printer engine, resulting in high quality, professionally looking printed documents.
- The same language can be also used for describing the screen output – the same code that is drawing to the screen can be used to draw to the printer without any translation.
- But, PostScript does not support multimedia, hyperlinks and some other things.
- In order to solve this problem, in the early 1990s Adobe Systems developed PDF.

Remarks on PDF

- PDF is a simplified version of the PostScript language, aimed far more at high quality on-screen display, document exchange, and hypertextual applications, than at printing quality.
- It has a very efficient font-embedding/replacement system to allow fonts to travel with the documents.
- It possesses a structured storage system to bundle these elements into a single file, with data compression where appropriate.
- A PDF document is usually more compact than the equivalent PostScript document, and has better defined structure.

- In addition to that, PDF is extended to enable to
  - use of intra- and inter-document hyperlinks
  - various dynamic effects for page transitions
  - working with forms
  - calling external applications, sound and video
  - embedding JavaScript programs into a PDF document
  - and many other things
- This extension is known as pdfmark.

- The pdfmark operator is a PostScript extension which is only implemented in Acrobat Distiller (as opposed to PostScript printers).
- Using this operator, many non-layout-related features of a PDF file (that have no counterpart in PostScript) can be defined in the original document or in the corresponding PostScript code.
- When PostScript files containing pdfmarks are processed by Acrobat Distiller, the corresponding PDF is generated.

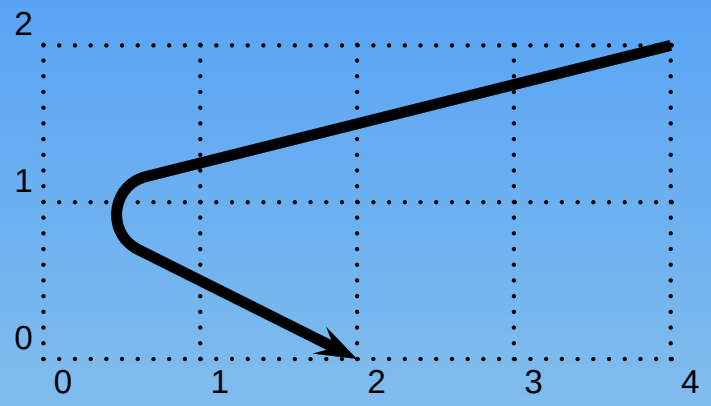
Examples

- In this section we will give a number of examples demonstrating various capabilities of TEX, PostScript (PSTrics) and PDF.
- The examples are obtained in the following way:
  - they are written in TEX, using PSTrics macro package,
  - TEX's `.dvi` output is then converted to PostScript using the `dvips` driver, and
  - the PostScript file is converted to PDF using GhostScript.
- We have tried to select the most characteristic examples.

Basic Graphics Objects

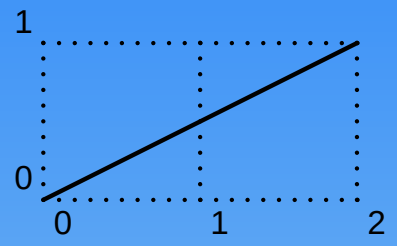
An example of a line connecting 3 points

(with a rounded corner)



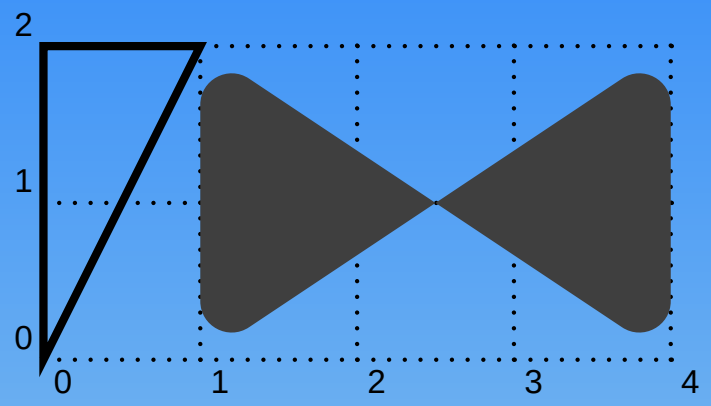

```
\psline[linewidth=2pt,linearc=.25]{->}(4,2)(0,1)(2,0)
```

An example of a simple line between two points



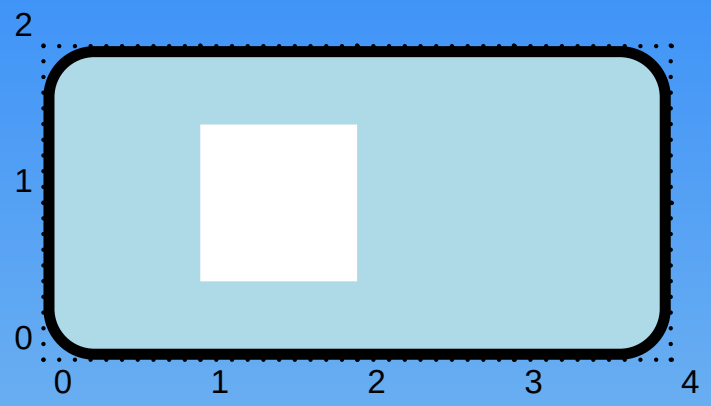

```
\qline(0,0)(2,1)
```

Two simple examples of polygons



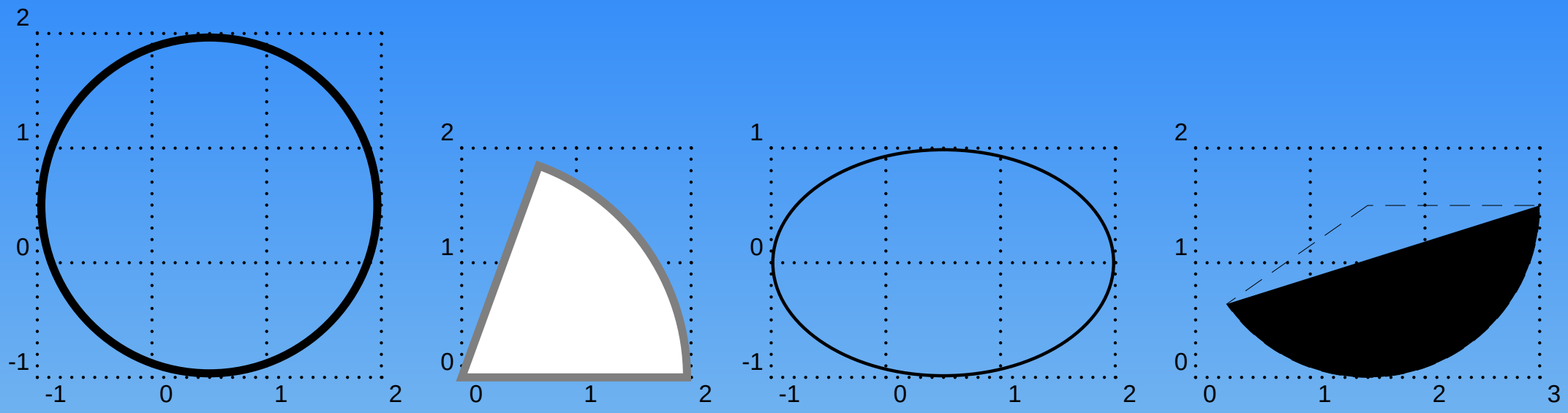

```
\pspolygon[linewidth=1.5pt](0,2)(1,2)
\pspolygon*[linearc=.2,linecolor=lightgray](1,0)(1,2)(4,0)(4,2)
```

Two examples of frames



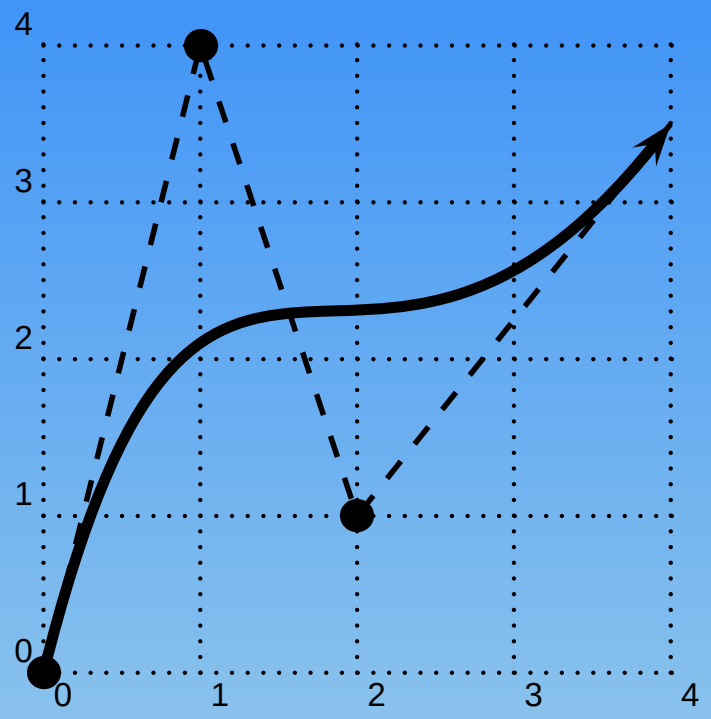

```
\psframe[linewidth=2pt,framearc=.3,fillstyle=solid,
  fillcolor=LightBlue](4,2)
\psframe*[linecolor=white](1,.5)(2,1.5)
```

Circle, Wedge, Ellipse and Arc



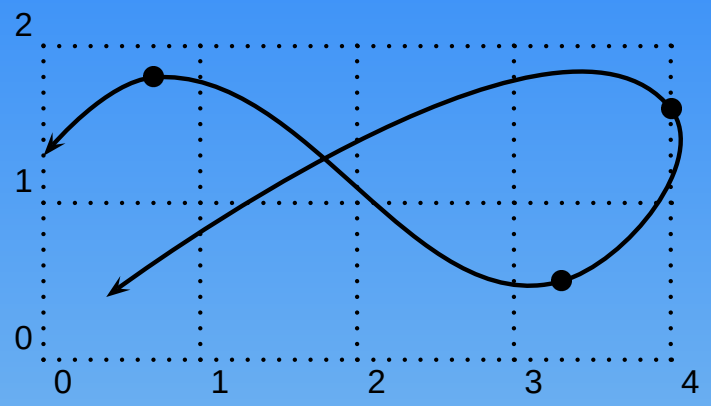

```
\pscircle[linewidth=2pt](.5,.5){1.5}
\pswedge[linecolor=gray,linewidth=2pt,fillstyle=solid]{2}{0}{70}
\psellipse[fillcolor=lightgray](.5,0)(1.5,1)
\psarc*[showpoints=true](1.5,1.5){1.5}{215}{0}
```

An example of a Bezier curve



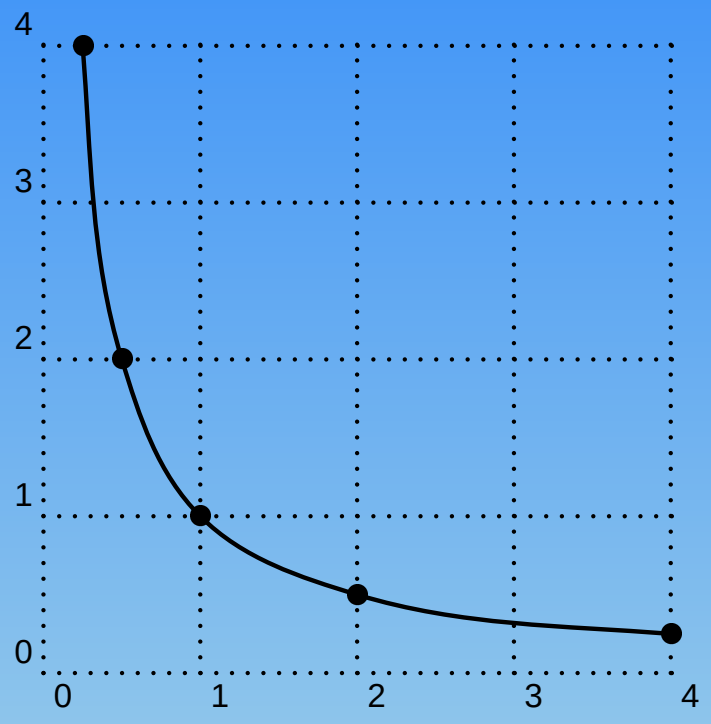

```
\psbezier[linewidth=2pt,showpoints=true]{->}(0,0)(1,4)(2,1)(4,3.5)
```

An example of an open curve interpolated through the points.



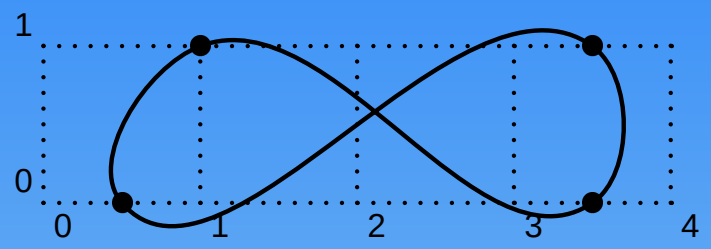

```
\pscurve[showpoints=true]{<->}(0,1.3)(0.7,1.8)(3.3,0.5)(4,1.6)(0.4,0.4)
```

Like the previous example, but the curve is not extended to the first and last points



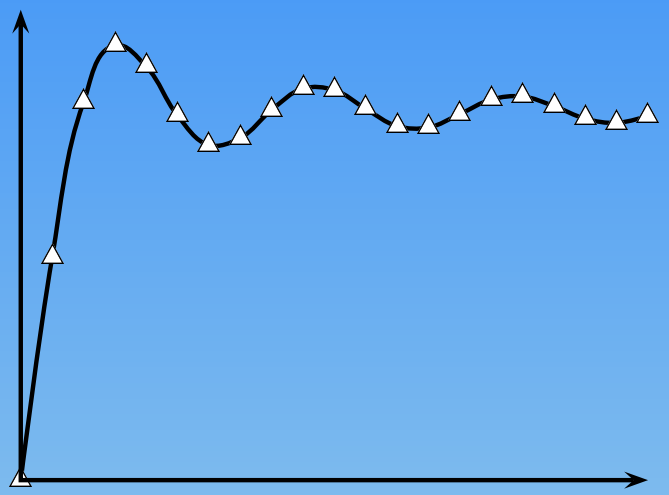

```
\psecurve[showpoints=true](.125,8)(.25,4)(.5,2)(1,1)(2,.5)(4,.25)(8,.125)
```

A closed curve through the points




```
\psccurve[showpoints=true](.5,0)(3.5,1)(3.5,0)(1,1)
```

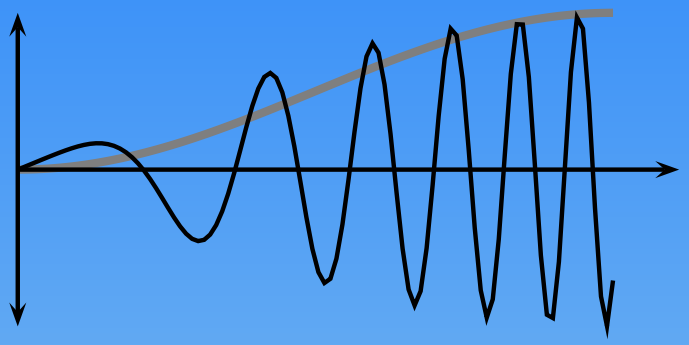
Plots – plotting lists of data generated by other programs



```
\psset{xunit=.2cm,yunit=1.5cm}
\savedata{\mydata}[
 {{0,0}, {1.,0.946083}, {2.,1.60541}, {3.,1.84865},
 {4.,1.7582},  {5.,1.54993}, {6.,1.42469}, {7.,1.4546},
 {8.,1.57419}, {9.,1.66504}, {10.,1.65835}, {11.,1.57831},
 {12., 1.50497}, {13.,1.49936}, {14.,1.55621}, {15.,1.61819},
 {16.,1.6313},  {17.,1.59014}, {18.,1.53661}, {19.,1.51863},
 {20.,1.54824}}]
\dataplot[plotstyle=curve,showpoints=true,
    dotstyle=triangle]{\mydata}
\psline{<->}(0,2)(0,0)(20,0)
```

This is a plot of the function Integral(sin(x)), and the data was generated by Mathematica

Plots – plotting a function $f(x)$ (expressed in PostScript)

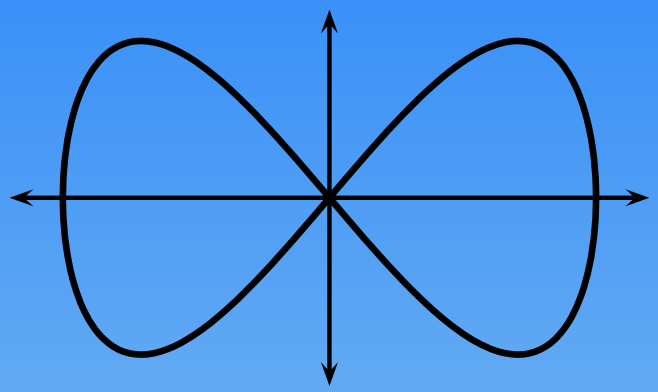


```
\psset{xunit=1.2pt}
  \psplot[linecolor=gray,linewidth=1.5pt,plotstyle=curve]%
      {0}{90}{x sin dup mul}
  \psplot[plotpoints=100]{0}{90}{x sin x 2 div 2 exp cos mul}
  \psline{<->}(0,-1)(0,1)
  \psline{->}(100,0)
```

These are the plots of the functions $\sin^2 x$ and $\sin x \cos((\frac{x}{2})^2)$

Note that PostScript is not designed for scientific computation, but it is good for graphing simple functions right from within TEX.

Parametric plots



```
\psset{xunit=1.2pt}
  \psset{xunit=1.7cm}
  \parametricplot[linewidth=1.2pt,plotstyle=ccurve]%
      {0}{360}{t sin t 2 mul sin}
  \psline{<->}(0,-1.2)(0,1.2)
  \psline{<->}(-1.2,0)(1.2,0)
```

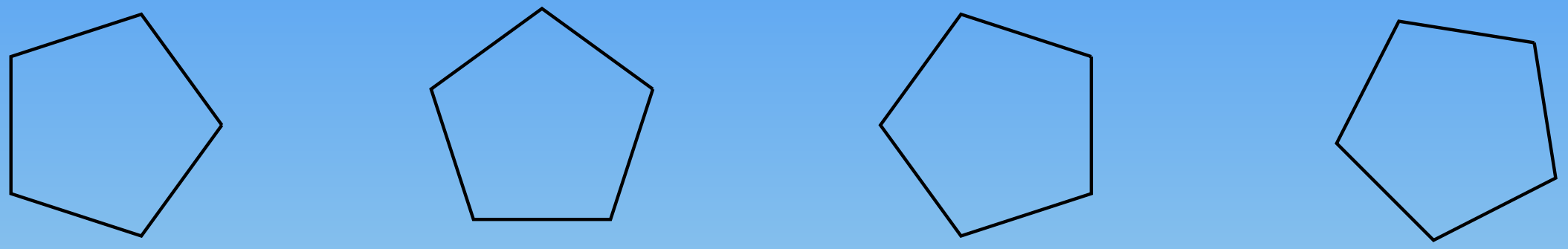
Here is a parametric plot of $(\sin t; \sin 2t)$

Complex Polygons

- **A PSTricks package** `pst-poly` **allow to draw easily various kinds of regular or non regular polygons, using the unique macro** `\PstPolygon`**, with various customization parameters.**
- **It is also a good example of the great power and flexibility of PSTricks, as in fact it is a very short program (it body is only 100 lines long) but nevertheless really powerful.**

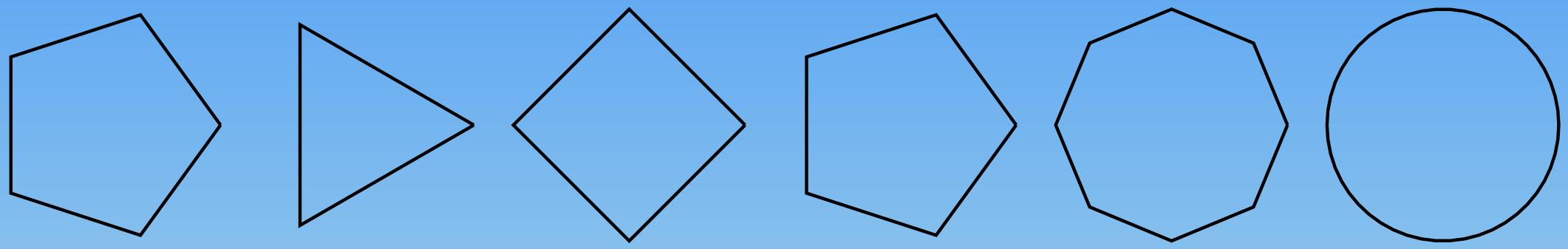
PolyRotation (real) : **rotation angle applied to the polygon (Default: 0 – no rotation).**

```
\PstPolygon  \hfill
\PstPolygon[PolyRotation=18]  \hfill
\PstPolygon[PolyRotation=36]  \hfill
\PstPolygon[PolyRotation=45]
```



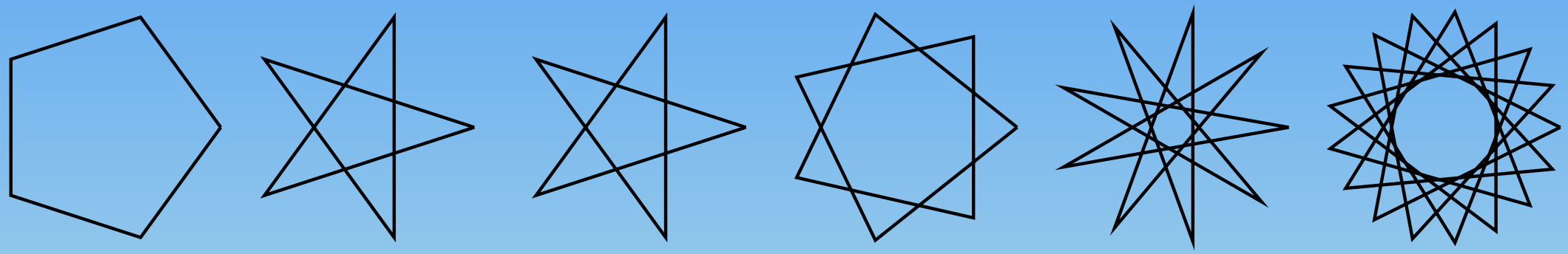
PolyNbSides (integer) : **number of sides of the polygon (Default: 5).**

```
\PstPolygon  \hfill
\PstPolygon[PolyNbSides=3]  \hfill
\PstPolygon[PolyNbSides=4]  \hfill
\PstPolygon[PolyNbSides=5]  \hfill
\PstPolygon[PolyNbSides=8]  \hfill
\PstPolygon[PolyNbSides=50]
```



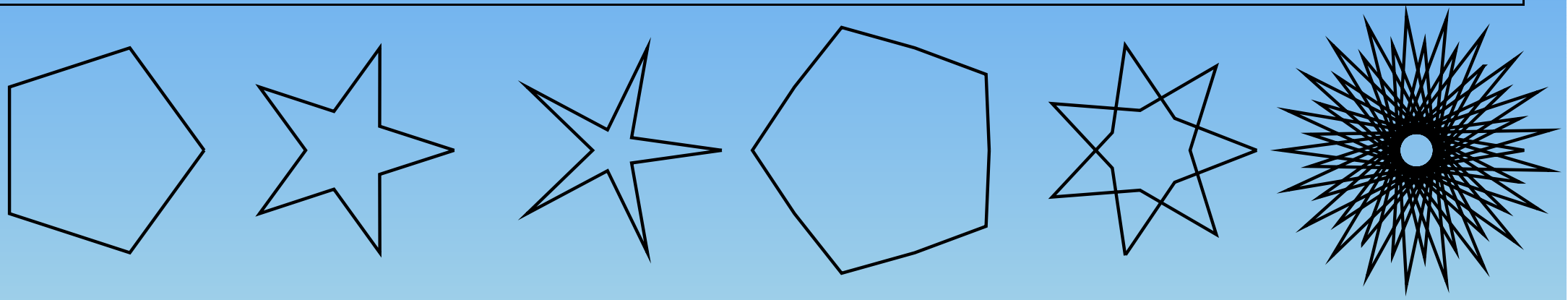
PolyOffset (integer) : **number of nodes to bypass to obtain each time the next one (Default: 1 — no node bypassed).**

```
\PstPolygon  \hfill
\PstPolygon[PolyOffset=2]  \hfill
\PstPolygon[PolyOffset=3]  \hfill
\PstPolygon[PolyNbSides=7,PolyOffset=2]  \hfill
\PstPolygon[PolyNbSides=9,PolyOffset=4]  \hfill
\PstPolygon[PolyNbSides=17,PolyOffset=6]
```



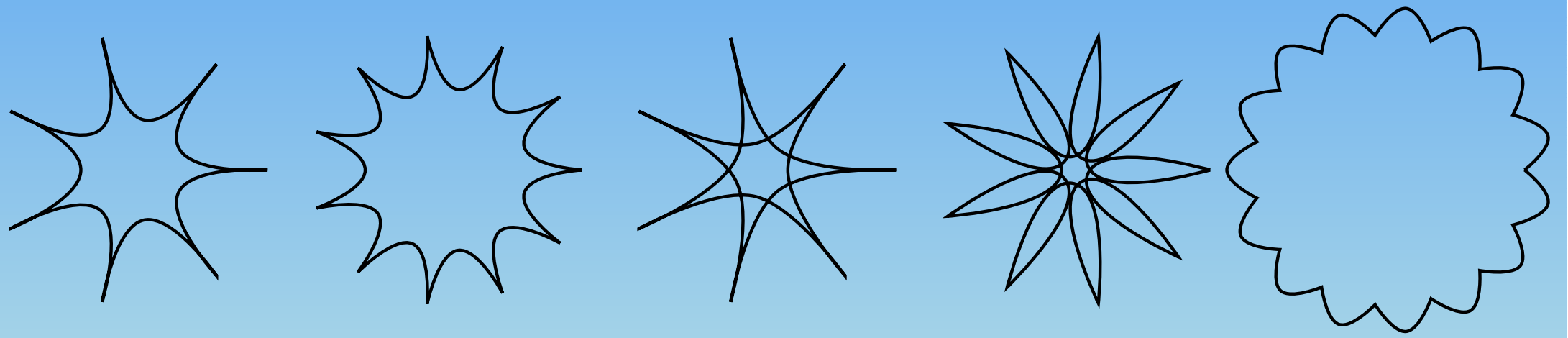
PolyIntermediatePoint (real) : **position of the intermediate point used to join each time the next node (Default: empty — not used).**

```
\PstPolygon  \hfill
\PstPolygon[PolyIntermediatePoint=0.38]  \hfill
\PstPolygon[PolyIntermediatePoint=0.2]  \hfill
\PstPolygon[PolyIntermediatePoint=1.2]  \hfill
\PstPolygon[PolyNbSides=7,PolyOffset=2,
            PolyIntermediatePoint=0.38]  \hfill
\PstPolygon[PolyNbSides=21,PolyOffset=2,
            PolyIntermediatePoint=-1.25]
```



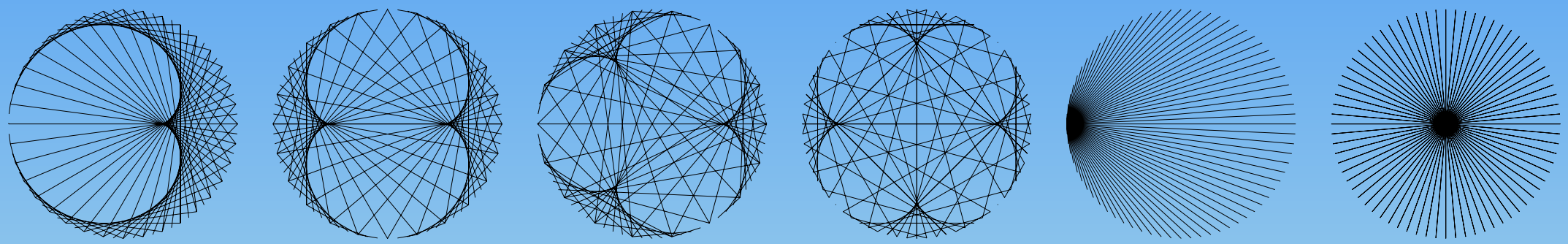
PolyCurves (boolean) : **boolean value to choose between straight line and curve to join each time the next node (Default: false – straight lines).**

```
\psset{PolyCurves=true}
\PstPolygon[PolyNbSides=7,PolyIntermediatePoint=0.38]\hfill
\PstPolygon[PolyNbSides=11,PolyIntermediatePoint=0.6]\hfill
\PstPolygon[PolyNbSides=7,PolyIntermediatePoint=0.2,
            PolyOffset=2]\hfill
\PstPolygon[PolyNbSides=9,PolyIntermediatePoint=0.1]\hfill
\PstPolygon[PolyNbSides=15,PolyIntermediatePoint=1.2]
```



PolyEpicycloid (boolean) : **boolean value to choose between polygon and epicycloid (Default: false — polygon).**

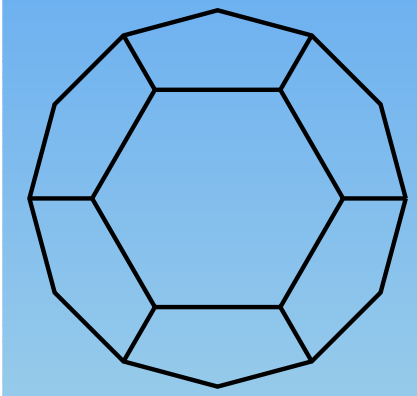
```
\psset{linewidth=0.001,PolyNbSides=72,PolyEpicycloid=true}
% Epicycloid of factor 1 is cardioid and of factor 2 nephroid
\multido{\i=2+1}{4}{\PstPolygon[PolyOffset=\i]\hfill}
\PstPolygon[PolyOffset=72]\hfill % Epicycloid of factor 71
\PstPolygon[PolyOffset=73]       % Epicycloid of factor 72
```



PolyName (string) : **name of the polygon, useful to have different names for the nodes of different polygons (Default: empty — no name).**

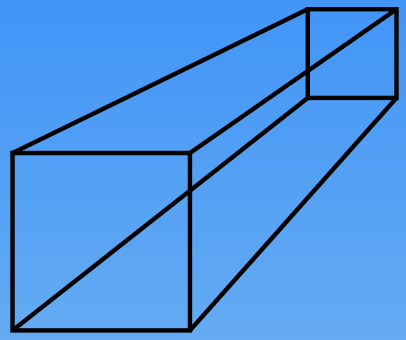
The center of the polygon has name `PolyName0` **and the nodes (vertices) have names** `PolyName1` **to** `PolyNameN`**.**

With this parameter, we can connect as we want nodes of different polygons:



```
\psset{PstPicture=false}
\begin{pspicture}(-1,-1)(1,1)
  \PstPolygon[unit=0.8,PolyName=A,PolyNbSides=6]
  \PstPolygon[unit=1.2,PolyName=B,PolyNbSides=12]
\end{pspicture}
\multido{\i=1+2}{6}{%
  \ncline{A\the\multidocount}{B\i}}
```

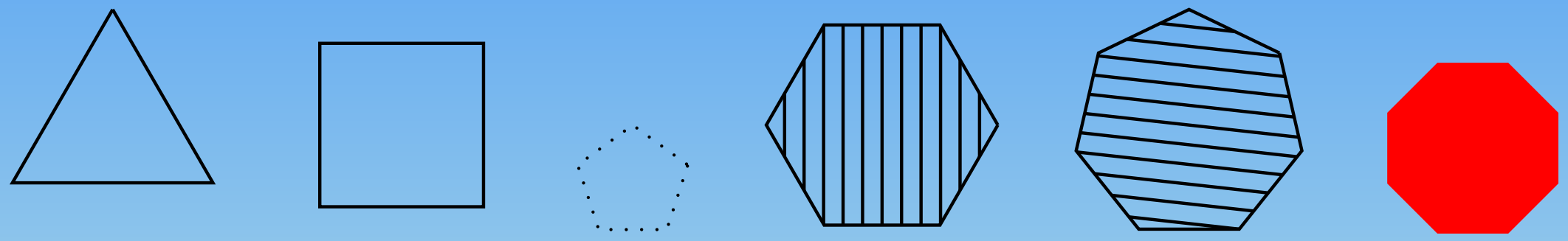
It is also a way (limited in fact...) to define three dimensional objects in perspective:



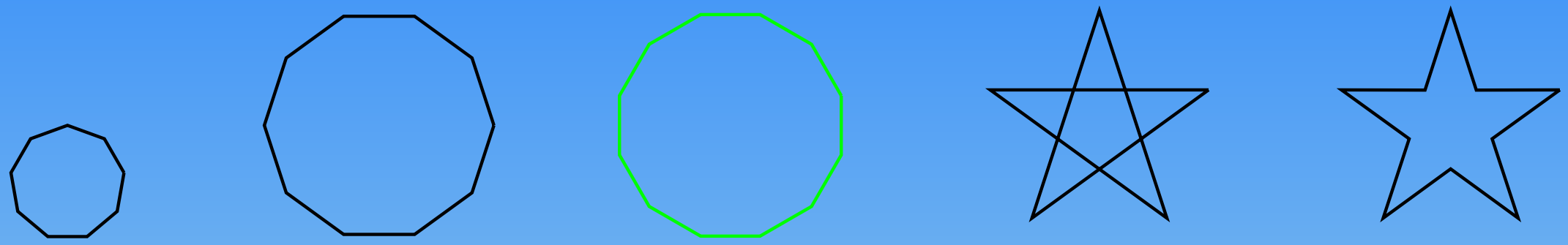
```
\psset{unit=0.8}
\begin{pspicture}(3,2.5)
  % \PstSquare is described later
  \rput[lb](0,0){\PstSquare[PolyName=A]}
  \rput[lb](2.5,2){\PstSquare[unit=0.5,
                              PolyName=B]}
  \multido{\i=1+1}{4}{\ncline{A\i}{B\i}}
\end{pspicture}
```

Some often used polygons and other related geometric objects are predefined, for immediate usage :

```
\PstTriangle  \hfill
\PstSquare  \hfill
\PstPentagon[unit=0.5,linestyle=dotted]  \hfill
\PstHexagon[fillstyle=hlines,hatchangle=90]  \hfill
\PstHeptagon[fillstyle=vlines]  \hfill
\PstOctogon*[unit=0.8,linecolor=red]
```

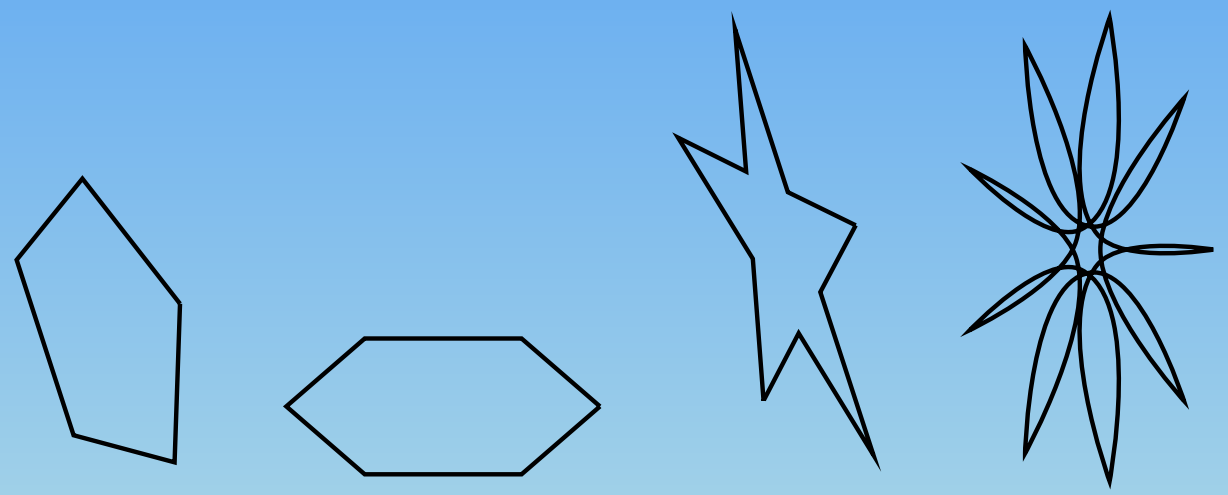


```
\PstNonagon[unit=0.5]  \hfill
\PstDecagon  \hfill
\PstDodecagon[linecolor=green]  \hfill
\PstStarFiveLines  \hfill
\PstStarFive
```



It is not so difficult to extend regular polygons to non regular ones, using a different value for horizontal and vertical units (nevertheless, the code is more tricky, as we must do all the trigonometry explicitely...)

```
\PstPentagon[xunit=0.5]  \hskip5mm
\PstHexagon[yunit=0.5]  \hskip5mm
\PstStarFive[xunit=0.5,yunit=1.5]  \hskip5mm
\PstPolygon[xunit=0.8,yunit=1.5,PolyNbSides=9,PolyOffset=2,
            PolyIntermediatePoint=0.1,PolyCurves=true]
```



Text Trics

Framed boxes

An example of a single framed box.

```
\psframebox[linewidth=1.5pt]{%
   \parbox[c]{5cm}{\raggedright An example
   of a single framed box.}}
```

A double frame is drawn with the gap between lines equal to `doublesep`

```
\psdblframebox[linewidth=1.5pt]{%
   \parbox[c]{5cm}{\raggedright A double frame
   is drawn with the gap between lines equal
   to {\tt doublesep}}}
```

Shadowed and circled boxes

Great Idea!!

```
\psshadowbox{\bf Great Idea!!}
```

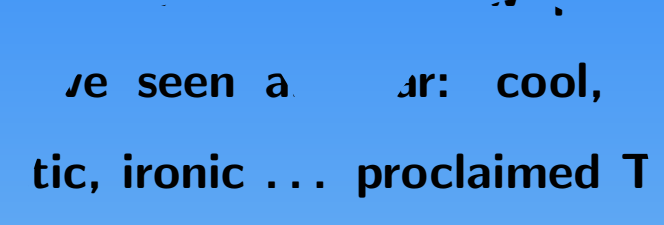
You are here

```
\pscirclebox{\begin{tabular}{c} You are \\
   here \end{tabular}}
```

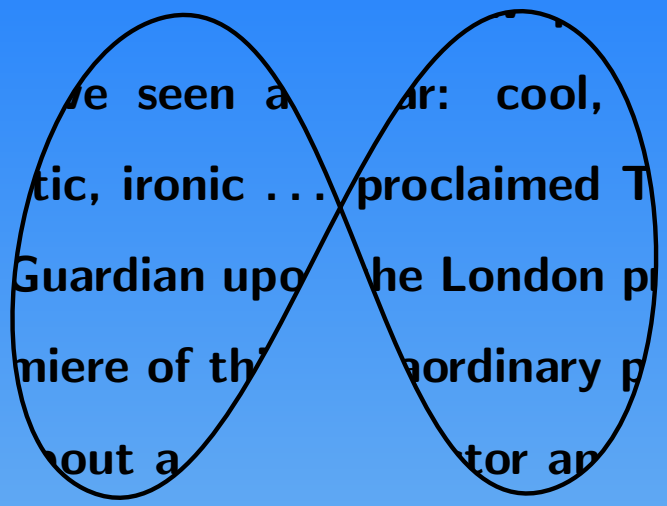
At the introductory price of $13.99, it pays to act now!

```
 At the introductory price of
\psovalbox[boxsep=false,linecolor=red]{\$13.99},
it pays to act now!
```

Clipping

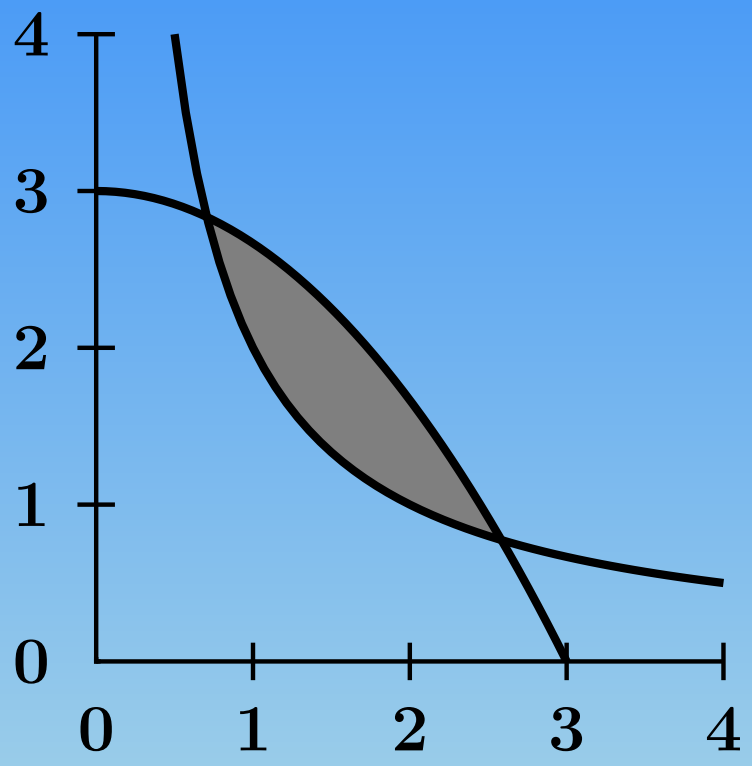



```
\parbox{4.5cm}{%
   \psclip{\psccurve[linestyle=none](.5,-3)(3.5,0)
       (3.5,-3)(1,0)}
   One of the best new plays I have seen all year: cool,
   poetic, ironic \ldots proclaimed {\em The Guardian} upon
   the London premiere of this extraordinary play about a
   Czech director and his actress wife, confronting exile
   in America.
   \endpsclip}
```



The same with `linestyle=solid`

Clipping – shading the region between two curves:




```
\psclip{%
   \pscustom[linestyle=none]{%
   \psplot{.5}{4}{2 x div}
   \lineto(4,4)}
   \pscustom[linestyle=none]{%
   \psplot{0}{3}{3 x x mul 3 div sub}
   \lineto(0,0)}}
   \psframe*[linecolor=gray](0,0)(4,4)
   \endpsclip
   \psplot[linewidth=1.5pt]{.5}{4}{2 x div}
   \psplot[linewidth=1.5pt]{0}{3}{3 x x mul 3 div sub}
   \psaxes(4,4)
```

Rotation and scaling boxes

Left Down Right

```
\rotateleft{Left} \rotatedown{Down} \rotateright{Right}
```

Big and long

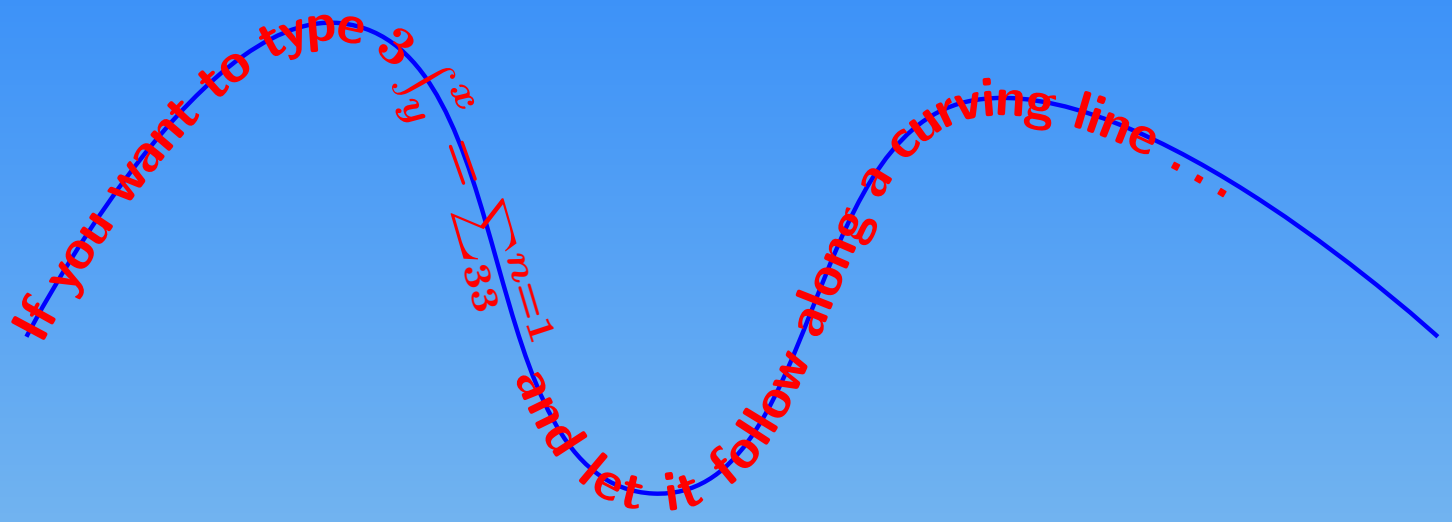
```
\scaleboxto(4,2){Big and long}
```

Question: How do Democrats organize a firing squad?

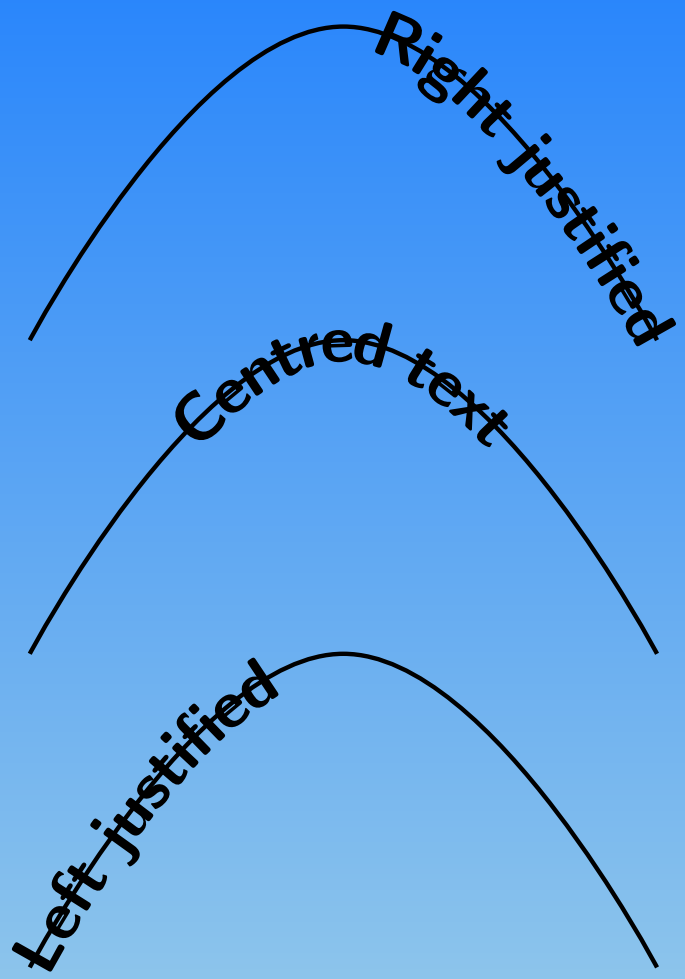
Answer: First they get in a circle, ...

```
Question: How do Democrats organize a firing squad?
\begin{Rotatedown}
\parbox{\hsize}{Answer: First they get in a circle,
\ldots\hss}%
\end{Rotatedown}
```

Writting Text Along a Path

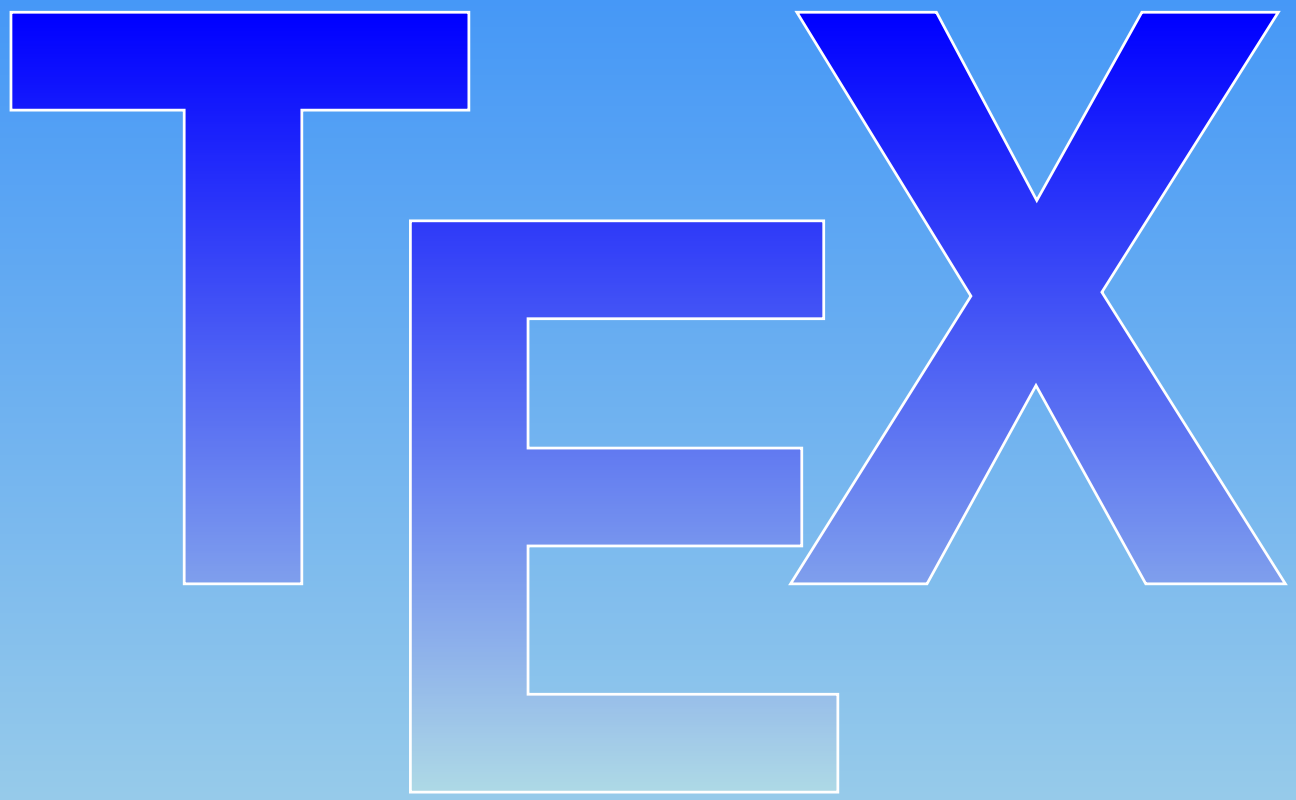


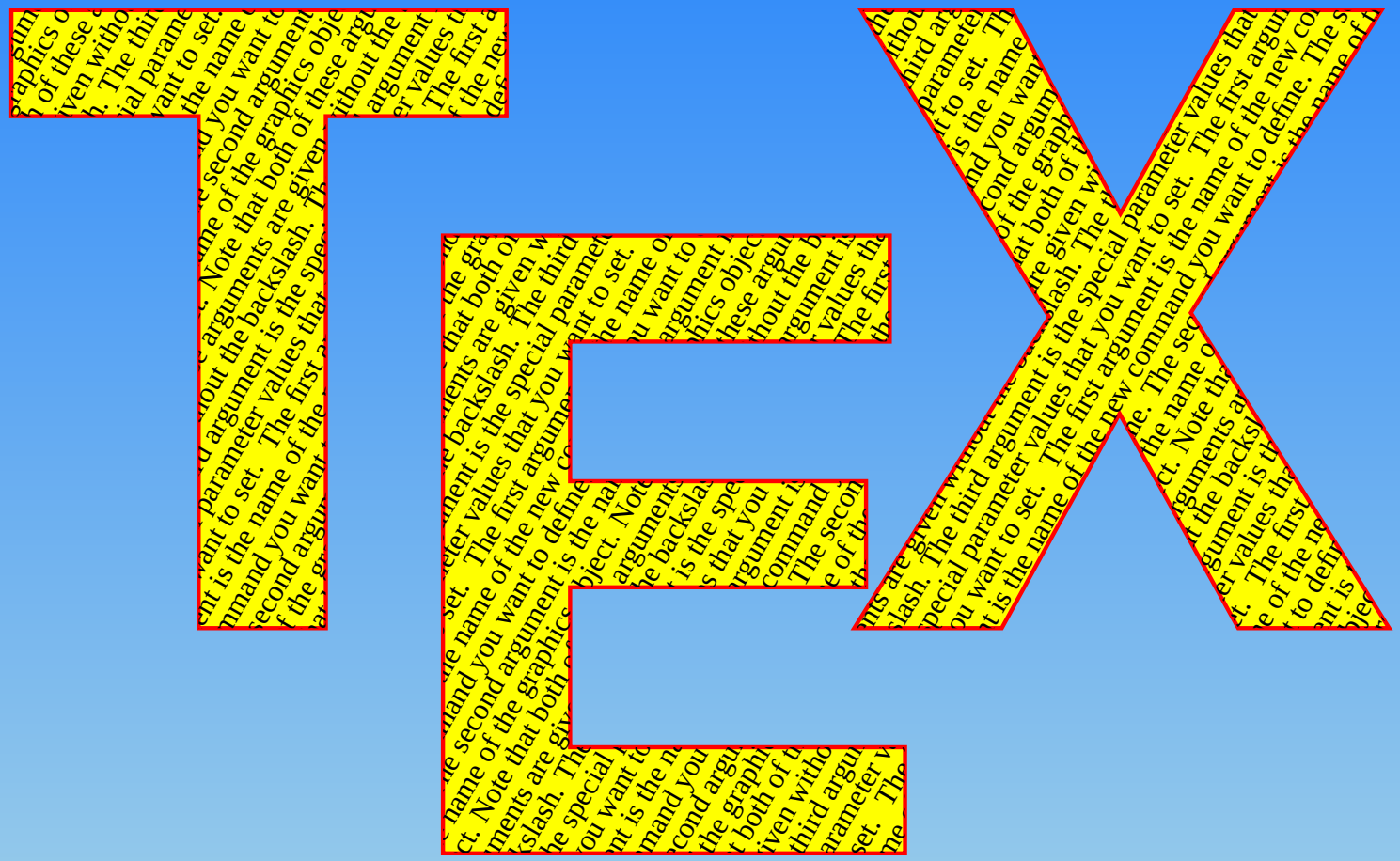

```
\begin{pspicture}(-4,-3.2)(3,0.2)
  \psset{linecolor=lightgray}
  \pstextpath{\pscurve(-4,-2)(-2,0)(0,-3)(2,-1)(3,-2)}
             {\color{red}
              If you want to type $3 \int^x_y = \sum^{n=1}_{33}$
              and let it follow along a curving line  \ldots}
\end{pspicture}
```




```
\begin{pspicture}(0,0)(4.2,6.2)
  \pstextpath[l]{\pscurve(0,0)(2,2)(4,0)}%
      {Left justified}
  \pstextpath[c]{\pscurve(0,2)(2,4)(4,2)}%
      {Centred text}
  \pstextpath[r]{\pscurve(0,4)(2,6)(4,4)}%
      {Right justified}
\end{pspicture}
```

Filling Character Outline




TEX

Nodes and Node Connections

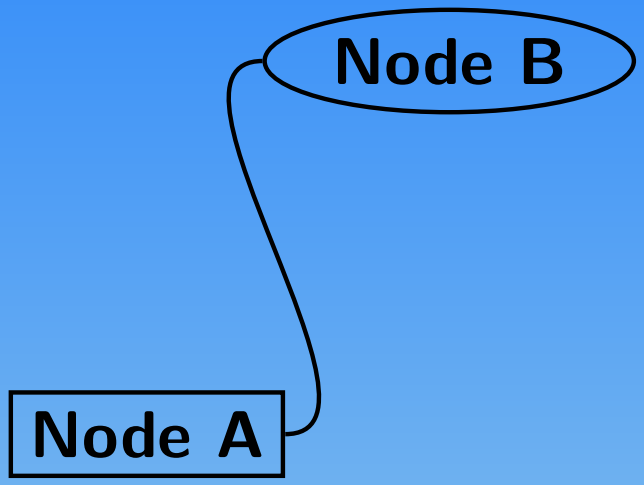
There are three components to the node macros:

Node definitions: The node definitions let you assign a name and shape to an object.

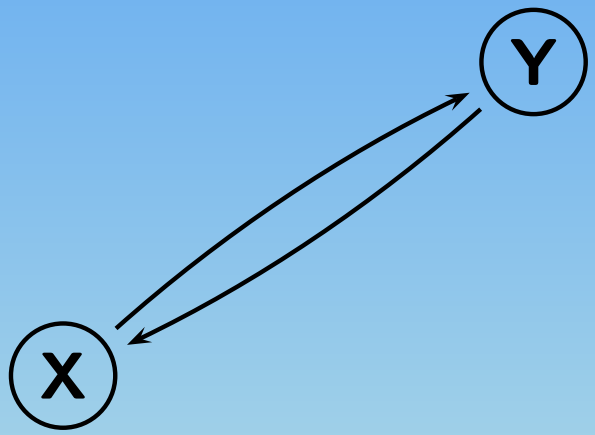
Node connections: The node connections connect two nodes, identified by their names.

Node labels: The node label commands let you affix labels to the node connections.

Node Connections



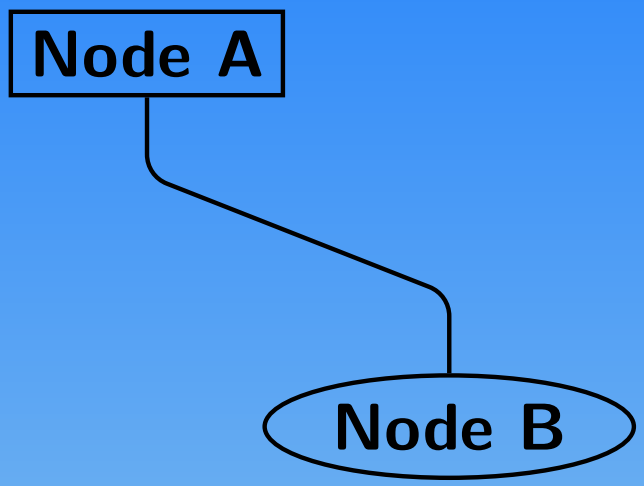

```
\rput[bl](0,0){\rnode{A}{\psframebox{Node A}}}
\rput[tr](4,3){\ovalnode{B}{Node B}}
\nccurve[angleB=180]{A}{B}
```



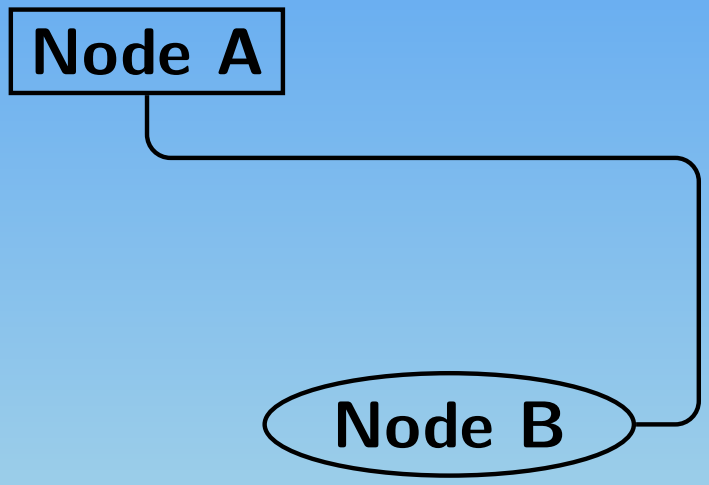

```
\cnodeput(0,0){A}{X}
\cnodeput(3,2){B}{Y}
\psset{nodesep=3pt}
\ncarc{->}{A}{B}
\ncarc{->}{B}{A}
```

Connect some words!

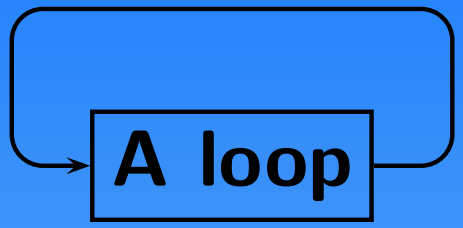
```
\rnode{A}{Connect} some \rnode{B}{words}!
\ncbar[nodesep=3pt,angle=-90]{<-**}{A}{B}
```




```
\rput[tl](0,3){\rnode{A}{\psframebox{Node A}}}
\rput[br](4,0){\ovalnode{B}{Node B}}
\ncdiag[angleA=-90,angleB=90,arm=.5,linearc=.2]{A}{B}
```

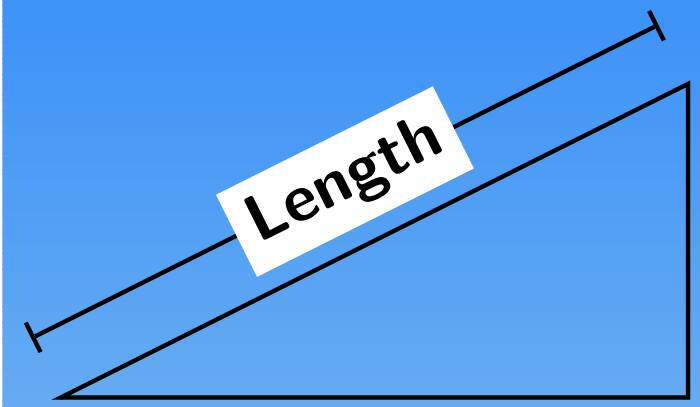



```
\rput[tl](0,3){\rnode{A}{\psframebox{Node A}}}
\rput[br](4,0){\ovalnode{B}{Node B}}
\ncangles[angleA=-90,arm=.4cm,linearc=.15]{A}{B}
```

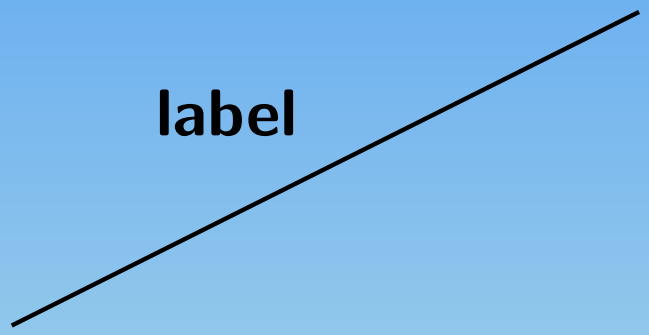



```
\rnode{a}{\psframebox{\Huge A loop}}
\ncloop[angleB=180,loopsize=1,arm=.5,linearc=.2]{->}{a}{a}
```

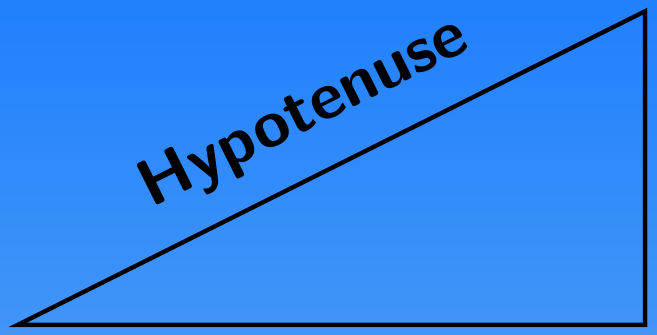
Attaching Labels



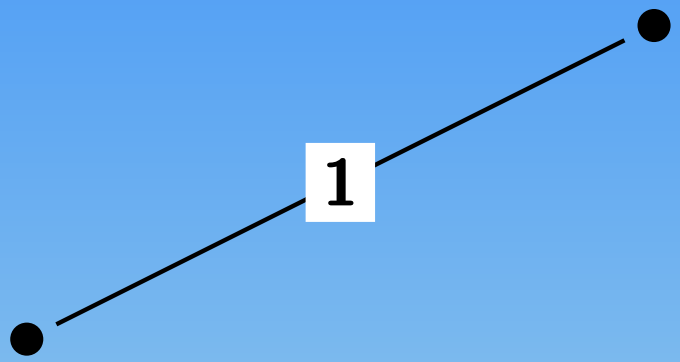

```
\pspolygon(0,0)(4,2)(4,0)
\pcline[offset=12pt]{|-|}(0,0)(4,2)
\lput*{:U}{Length}
```




```
\pcline(0,0)(4,2)
\lput{:U}{\rput[r]{N}(0,.4){label}}
```

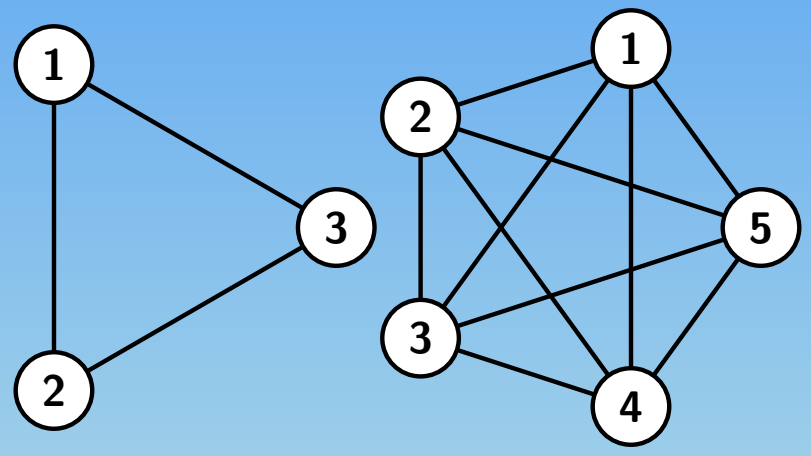


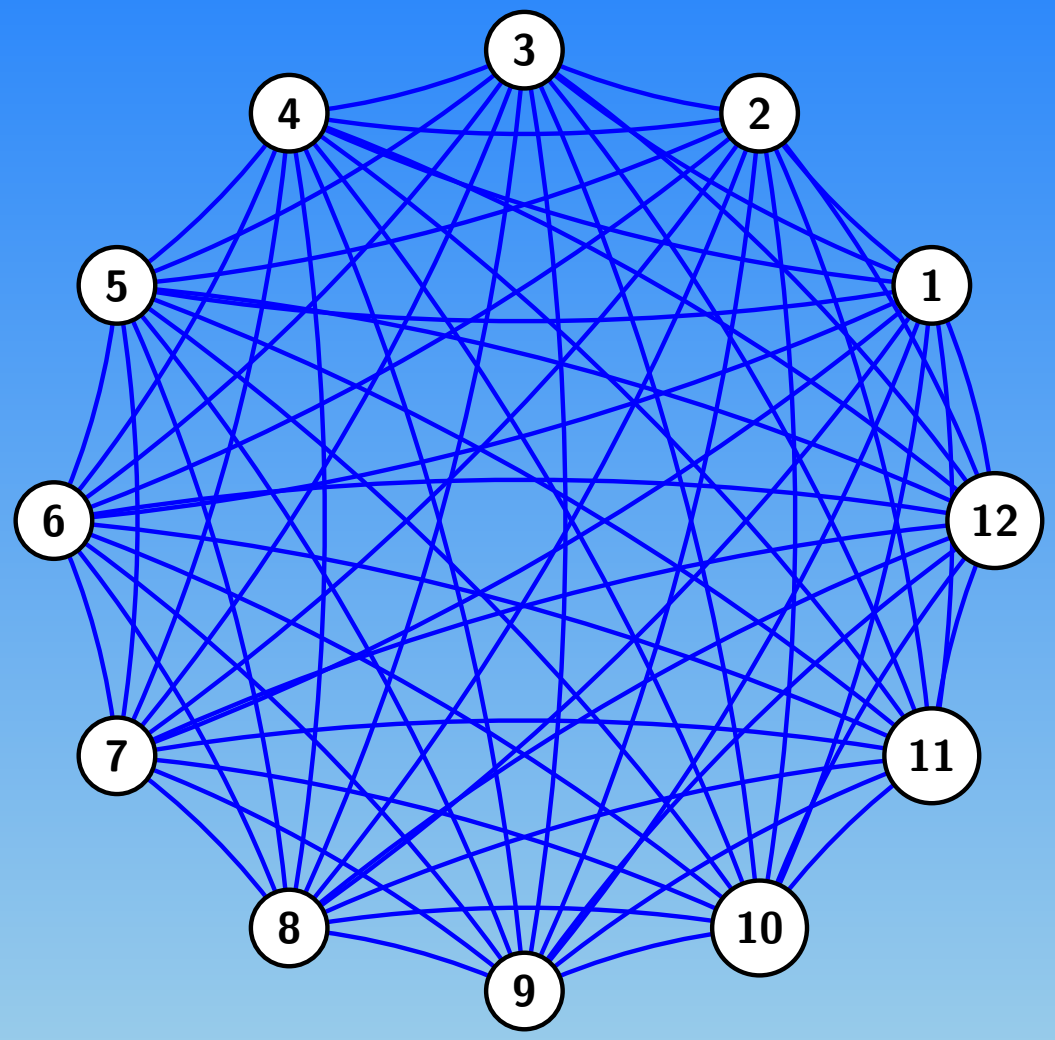

```
\pspolygon(0,0)(4,2)(4,0)
\pcline[linestyle=none](0,0)(4,2)
\aput{:U}{Hypotenuse}
```




```
\cnode*(0,0){3pt}{A}
\cnode*(4,2){3pt}{B}
\ncline[nodesep=3pt]{A}{B}
\mput*{1}
```

Nodes and node connections and labels are very useful in drawing graphs.

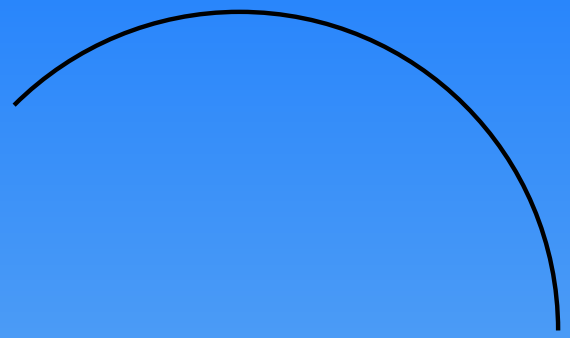




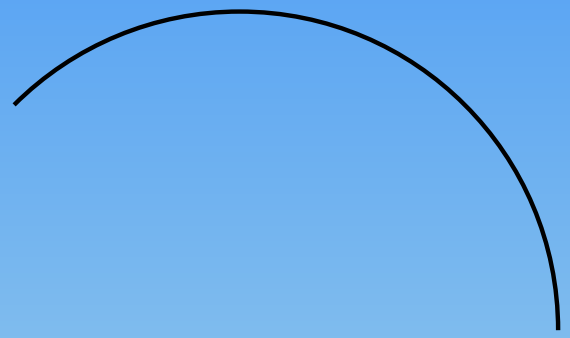
Special Coordinates

The command `\SpecialCoor` **enables a special feature that lets us specify coordinates in a variety of ways, in addition to the usual Cartesian coordinates.**

We can use polar coordinates, relative coordinates etc.



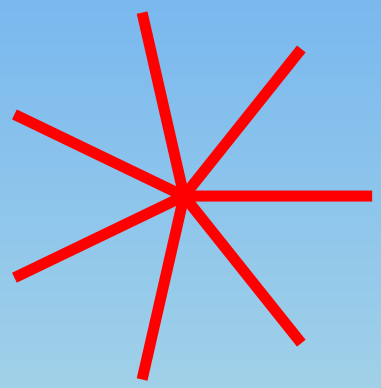
```
\psarc(0,0){.8in}{0}{(-1,1)}
```



```
\SpecialCoor
\psarc(0,0){.8in}{0}{135}
```

Angles, in polar coordinates and other arguments, should be a number giving the angle in degrees, by default.

We can also change the units used for angles with the command `\degrees[num]`**, where** `num` **should be the number of units in a circle.**



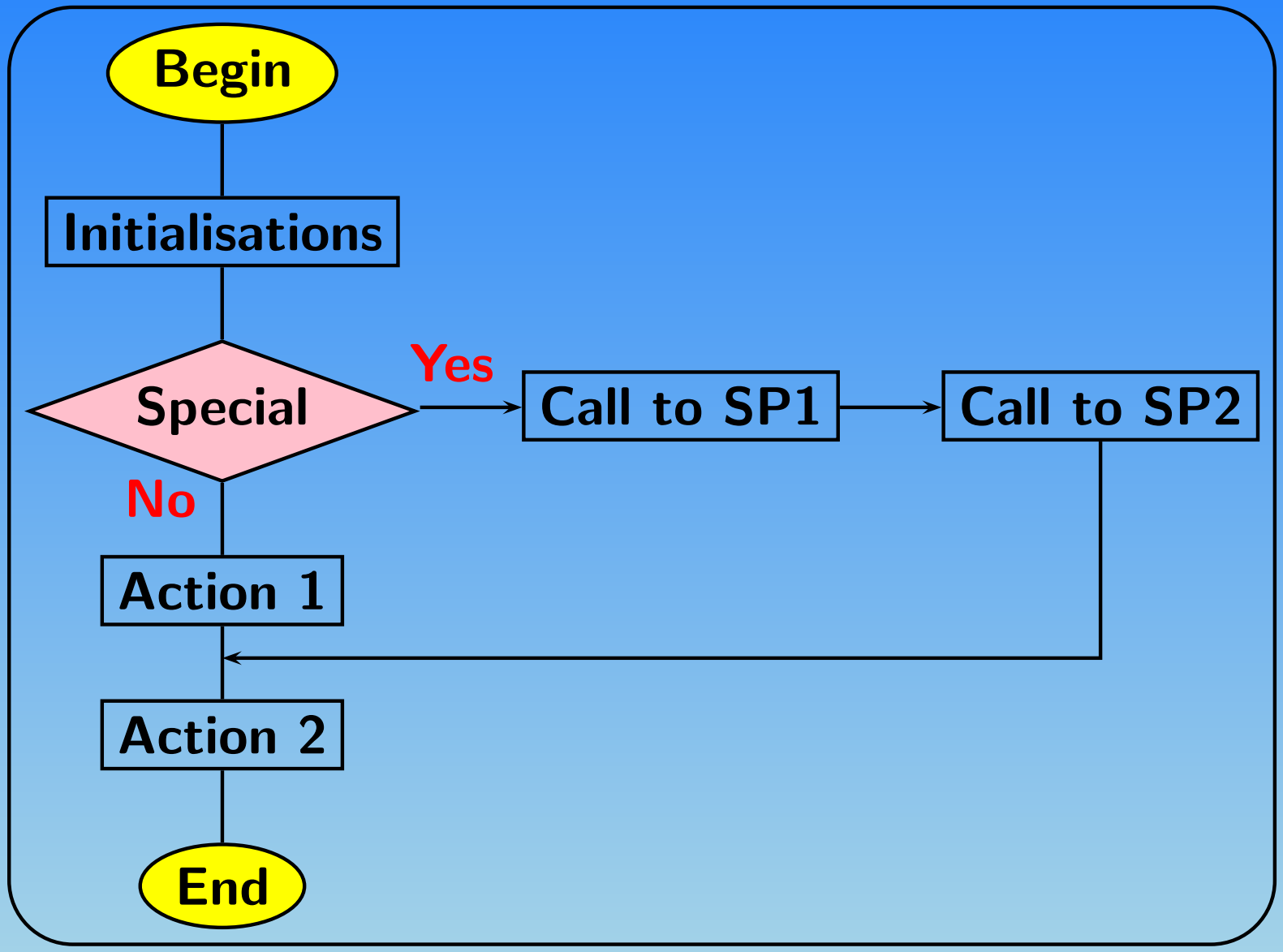
```
\pspicture(-1.2,-1.2)(1.2,1.2)
\psset{linewidth=2pt, linecolor=red}
\SpecialCoor
    \degrees[70]
    \psline(0,0)(1.2;0)
    \psline(0,0)(1.2;10)
    \psline(0,0)(1.2;20)
    \psline(0,0)(1.2;30)
    \psline(0,0)(1.2;40)
    \psline(0,0)(1.2;50)
    \psline(0,0)(1.2;60)
\NormalCoor
\endpspicture
```

Overlays

- Overlays are mainly of interest for making slides, and the overlay macros described in this section are mainly of interest to TEX macro writers who want to implement overlays in a slide macro package.
- For example, the `seminar.sty` package, a LATEX style for notes and slides, uses PSTricks to implement overlays.
- Overlays are made by creating an `\hbox` and then outputting the box several times, printing different material in the box each time.

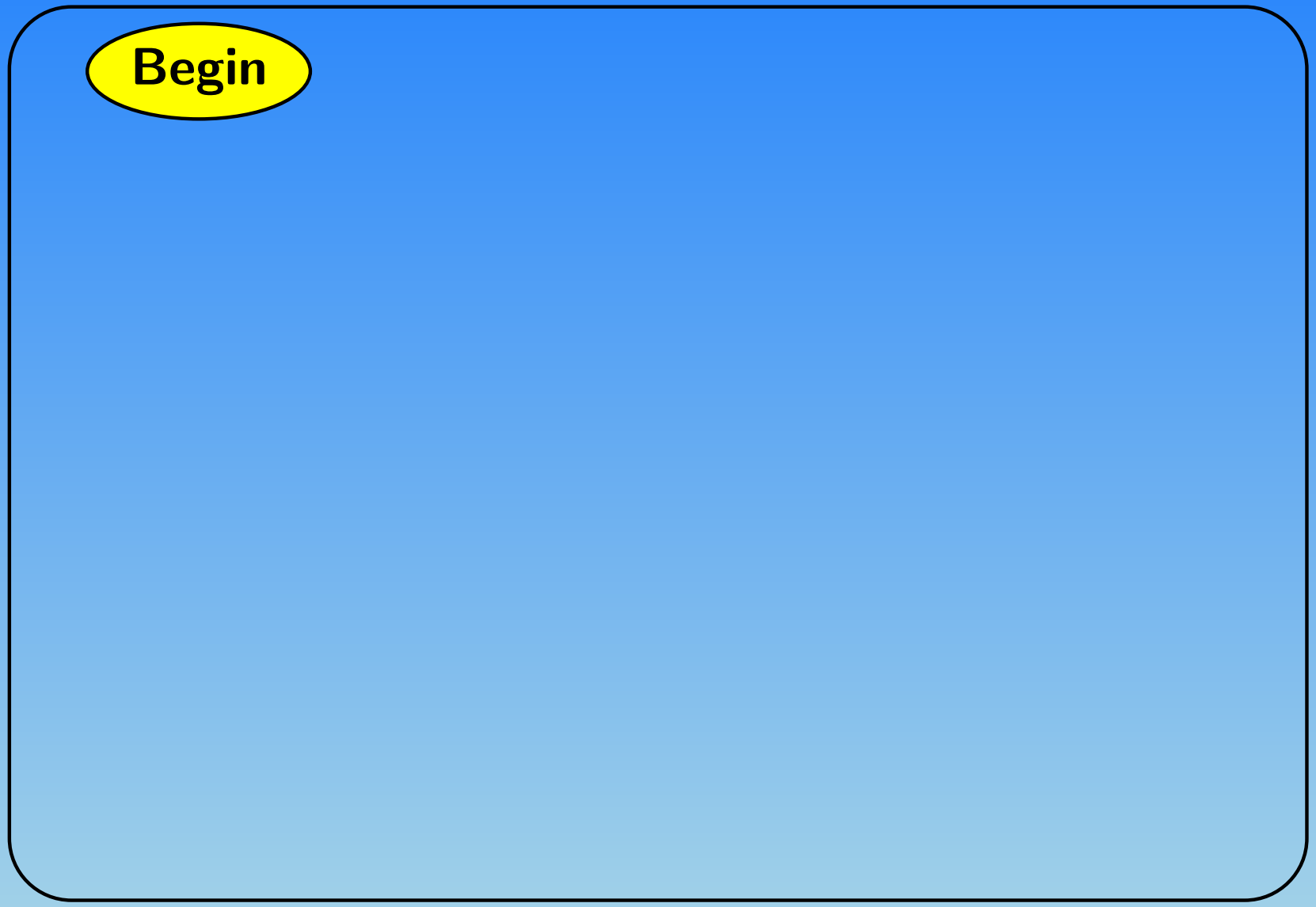
- **We distinguish between two kinds of overlays:**
  - **Cumulative overlays – with each new one added to the preceding.**
  - **Progressive overlays – where the first slide is kept but each new one will replace the preceding.**

Flow diagram using nodes and node connections

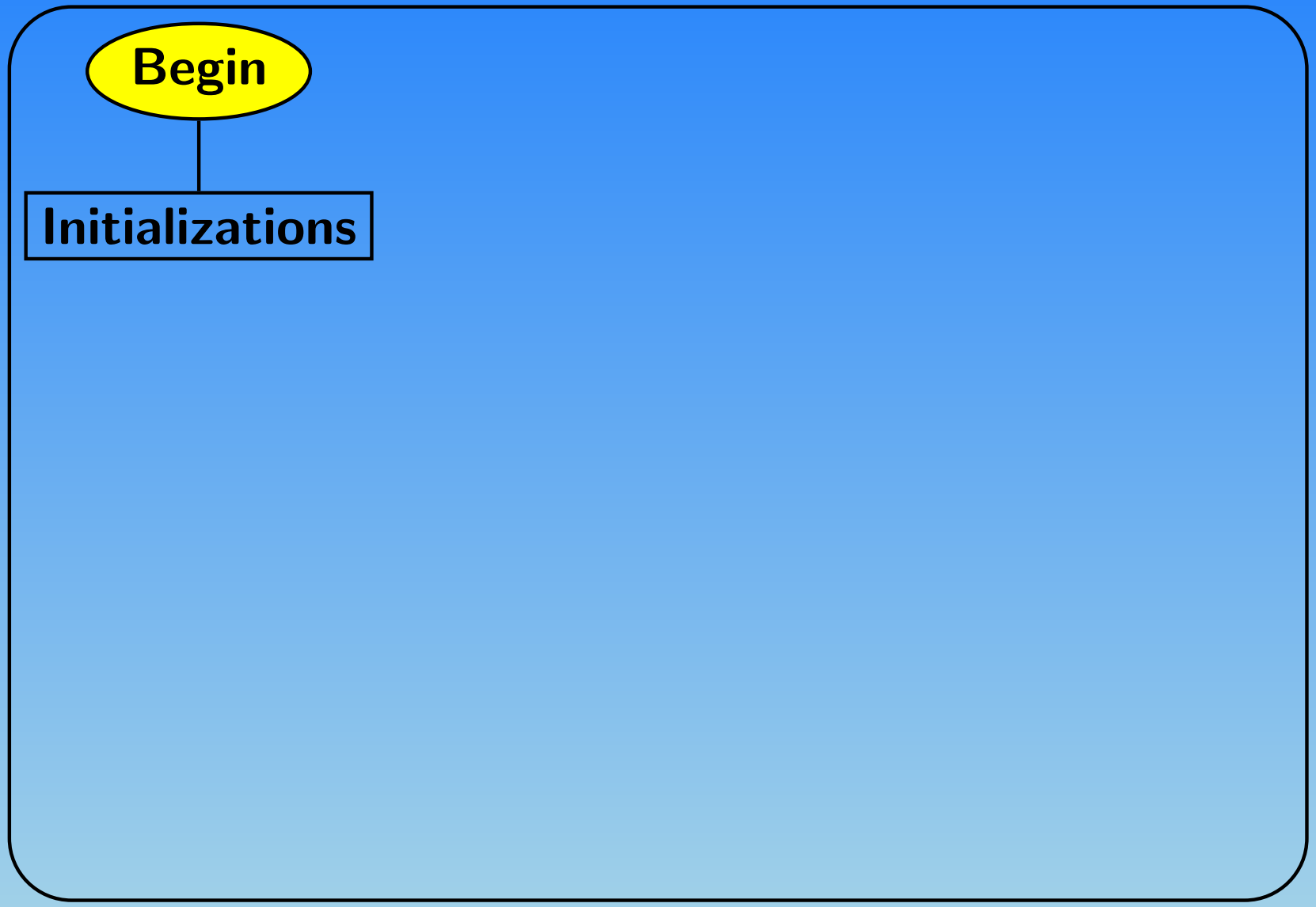


The same diagram with overlays

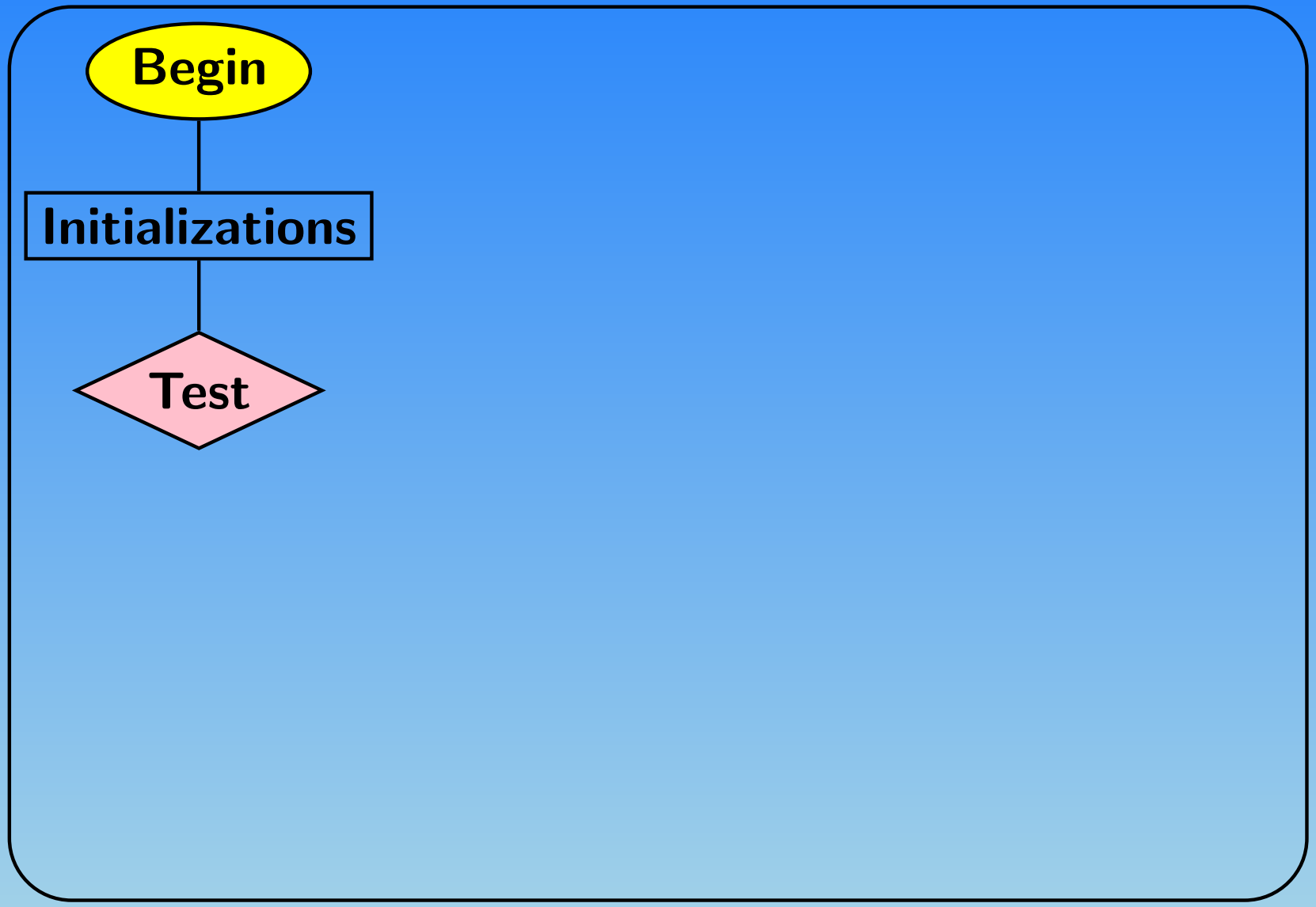
The same diagram with overlays



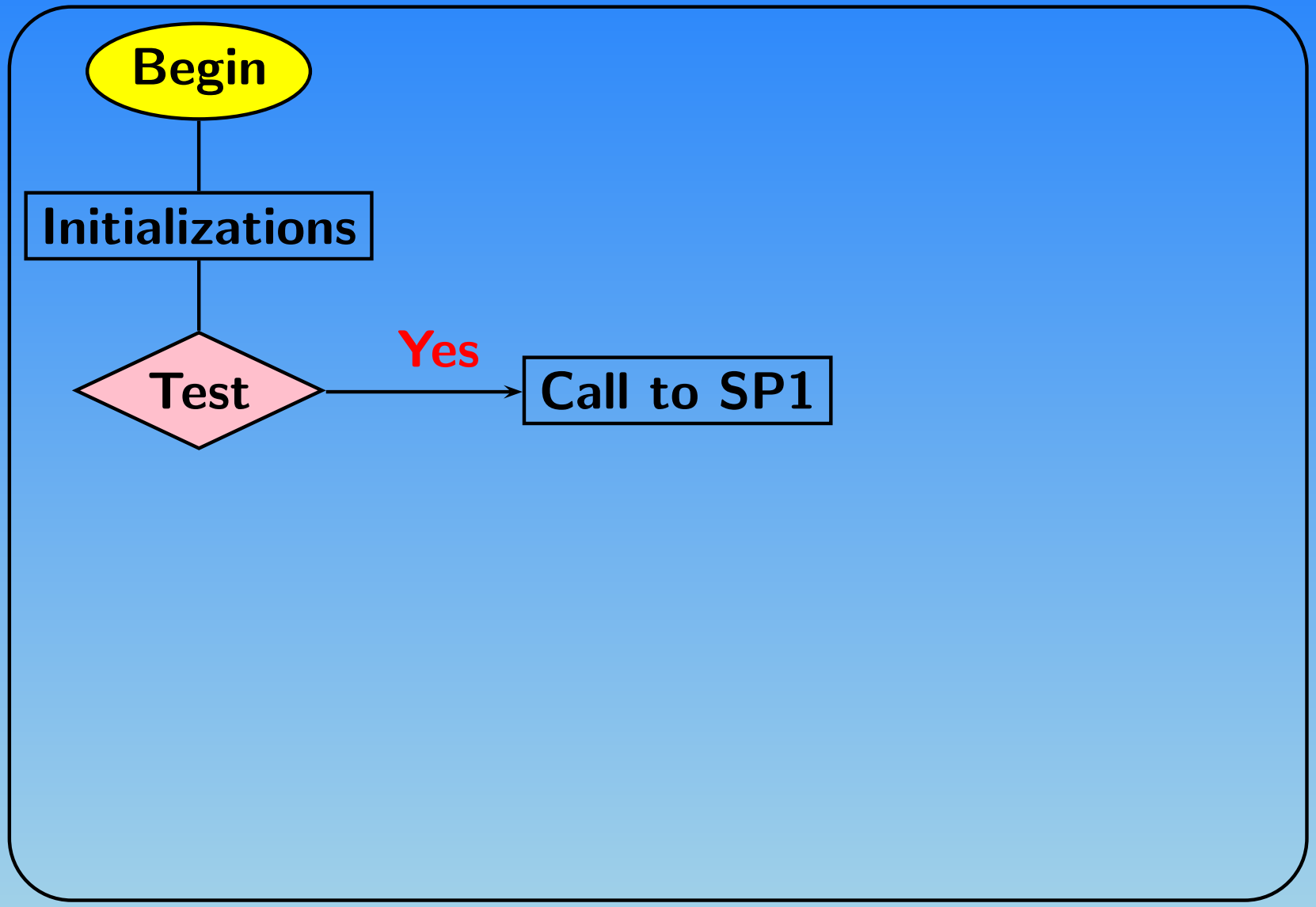
The same diagram with overlays



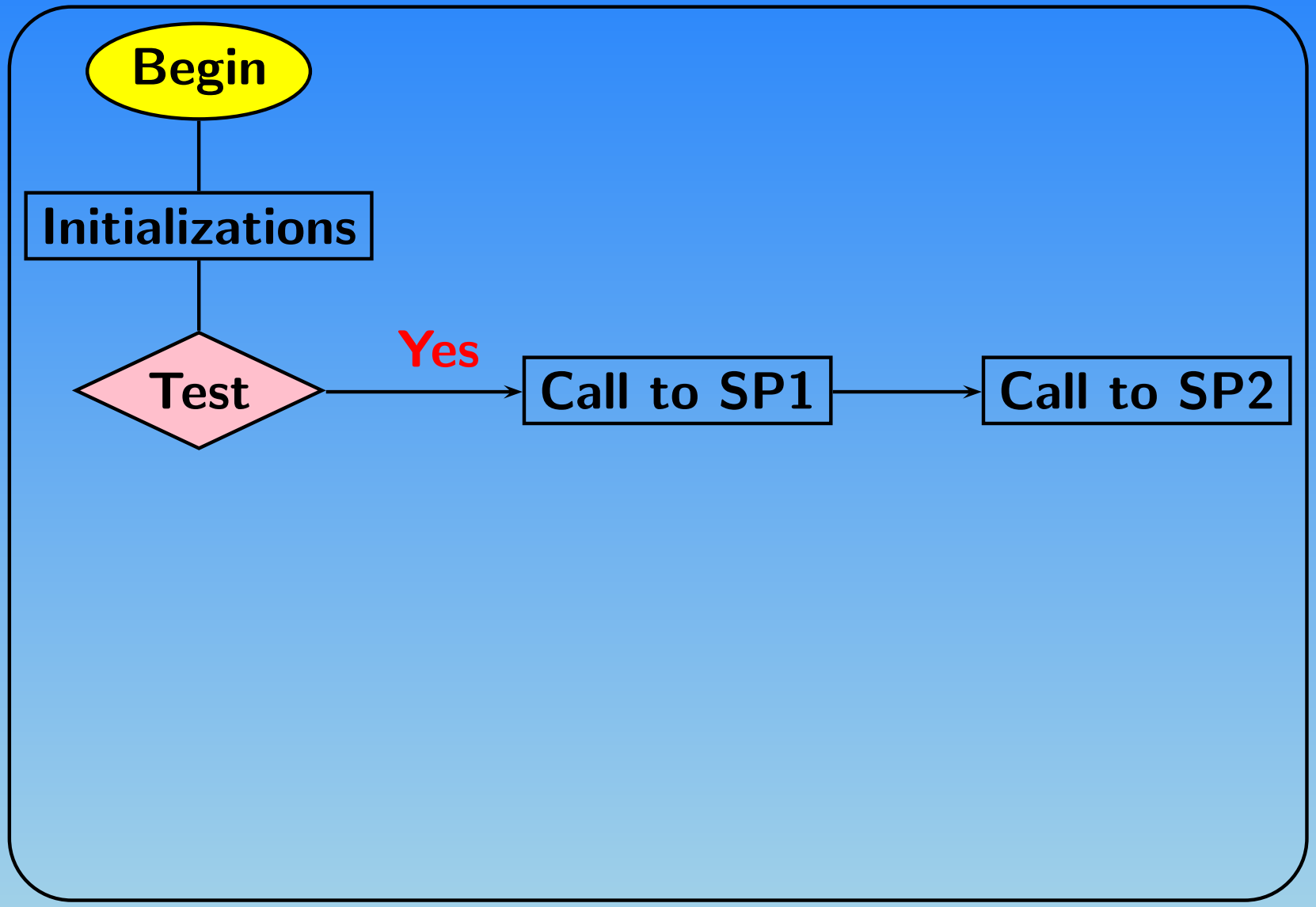
The same diagram with overlays



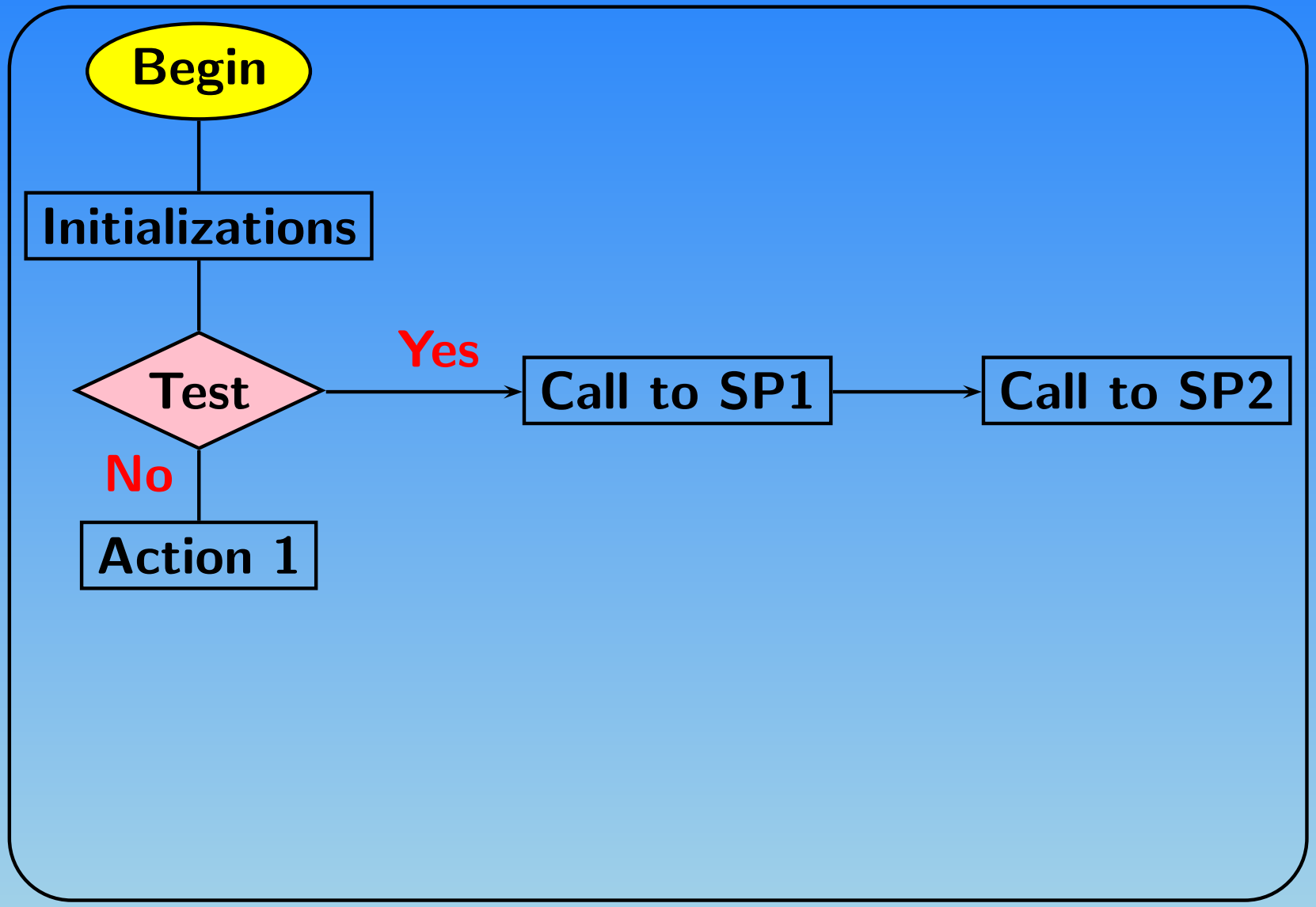
The same diagram with overlays



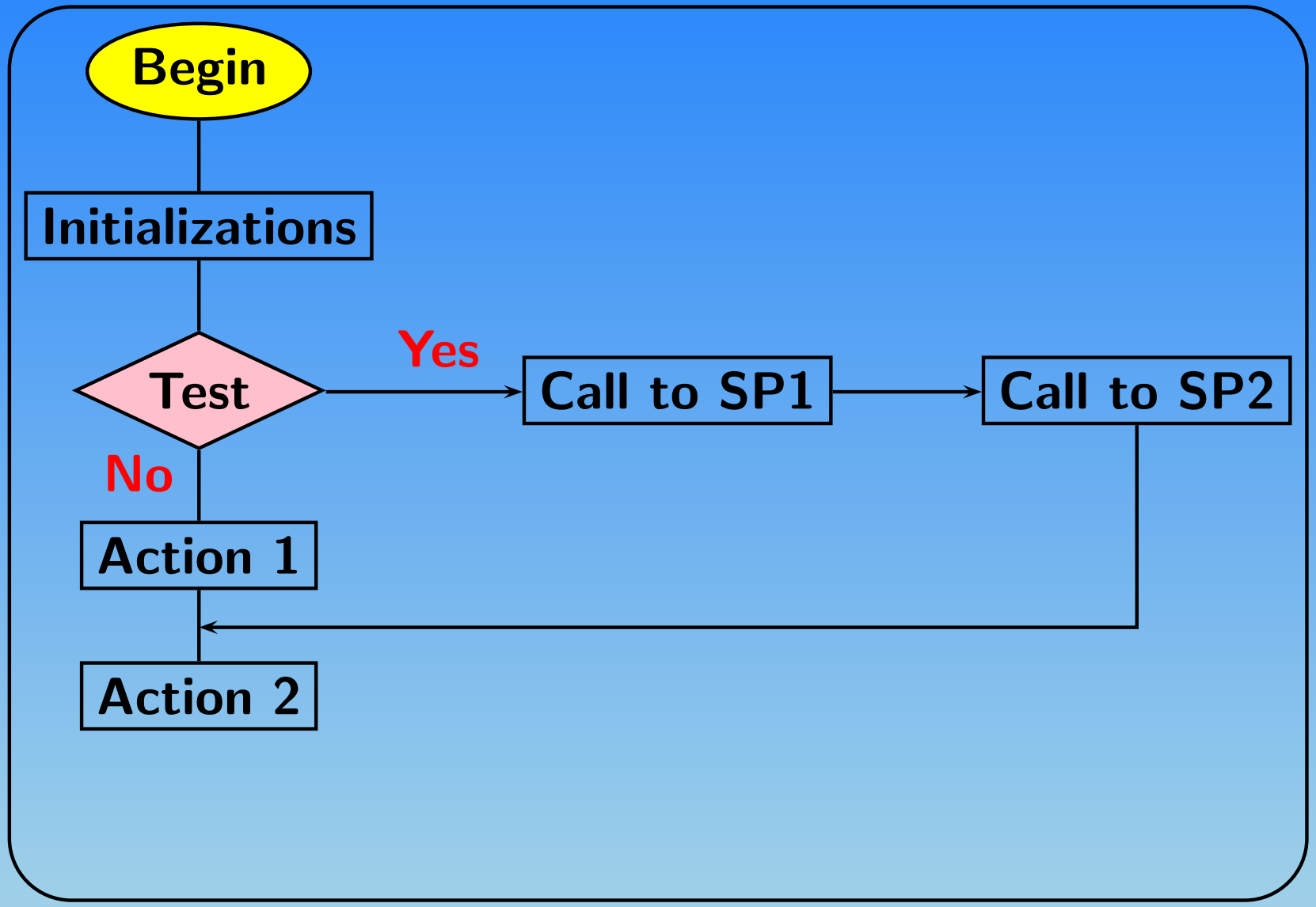
The same diagram with overlays



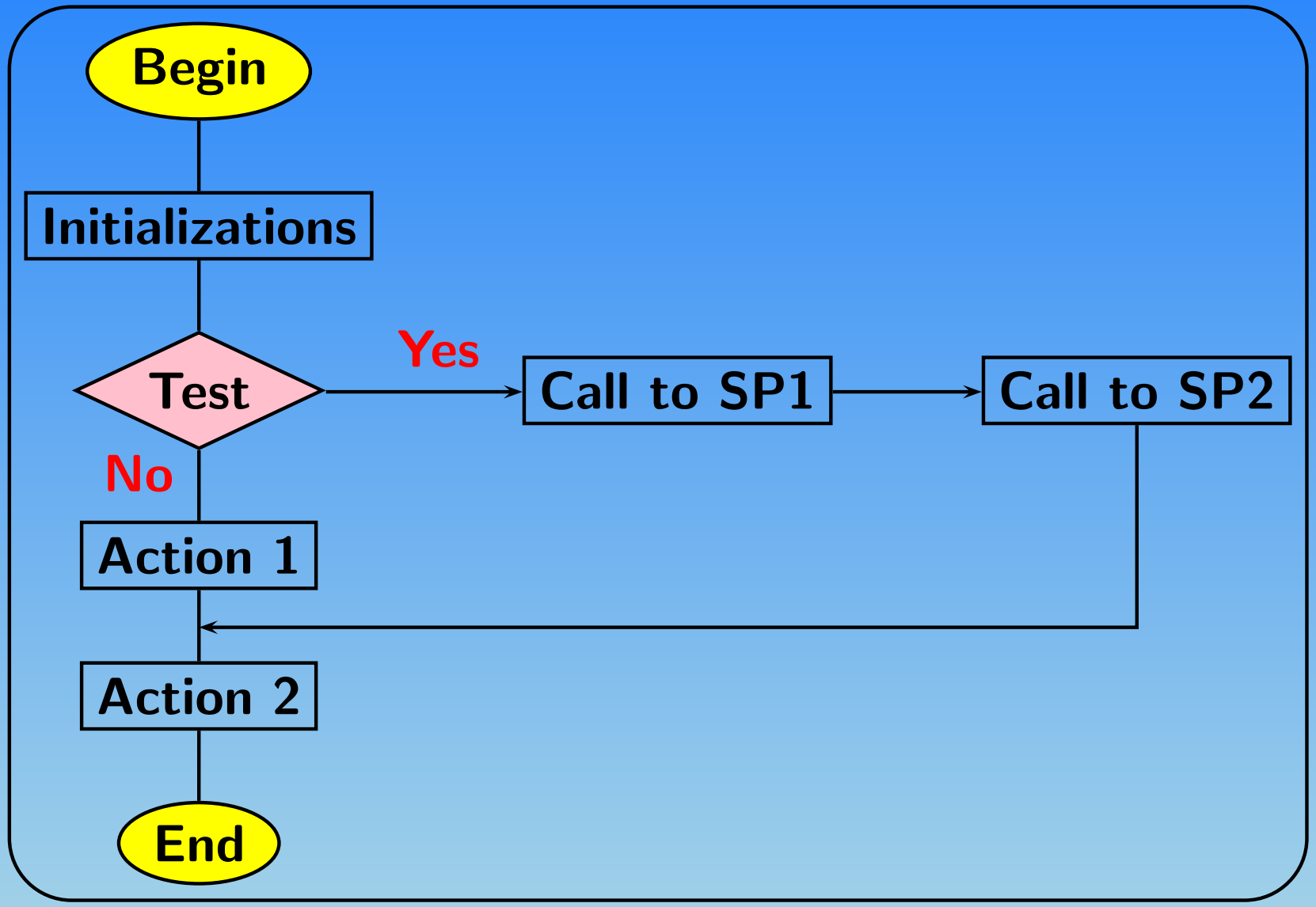
The same diagram with overlays



The same diagram with overlays



The same diagram with overlays



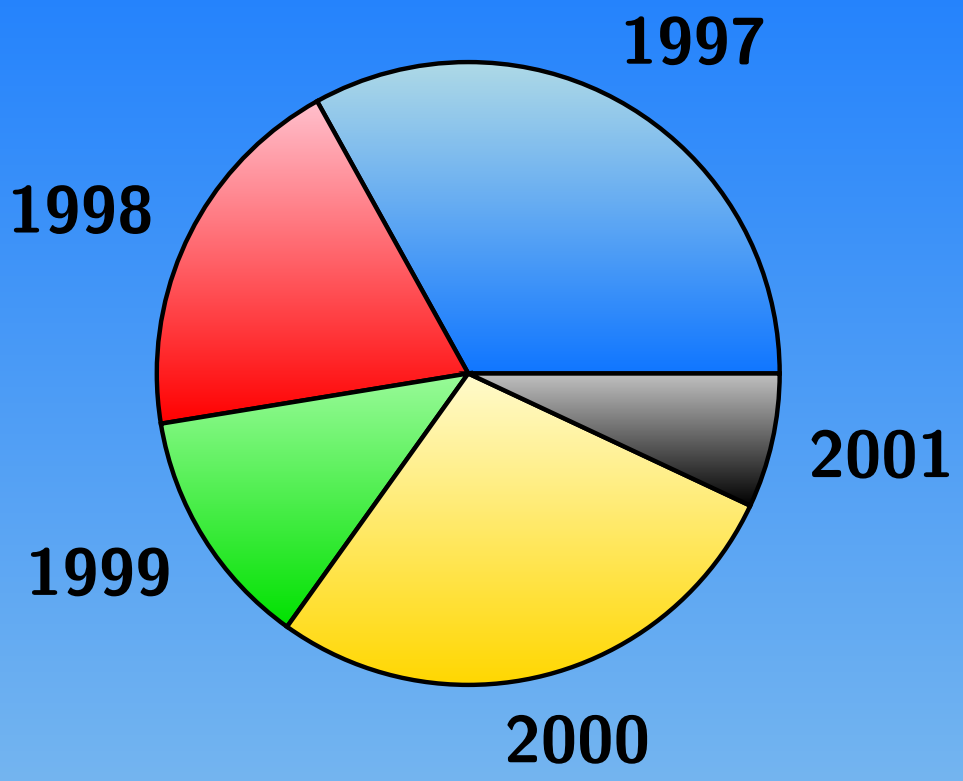
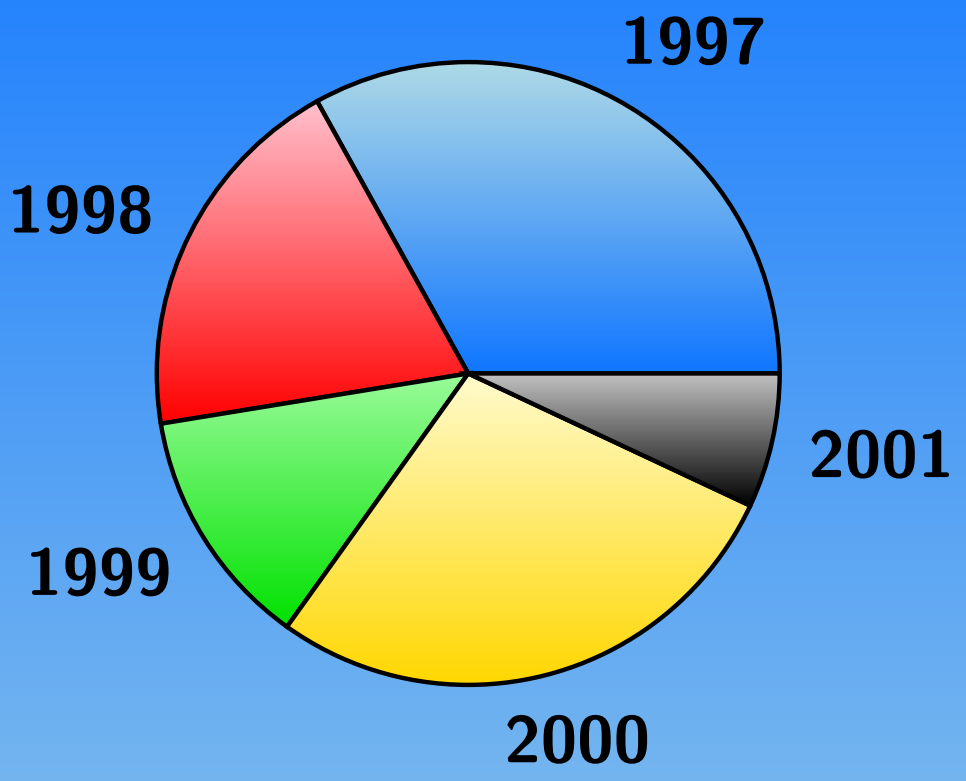


Figure 1: Results of the last five years



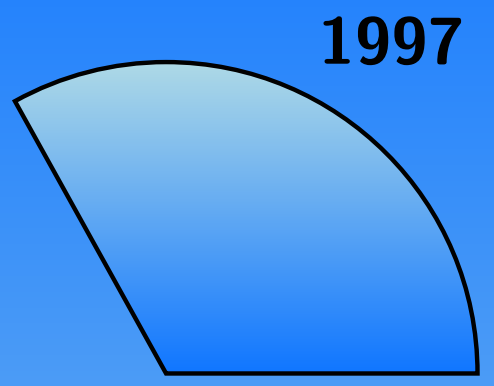
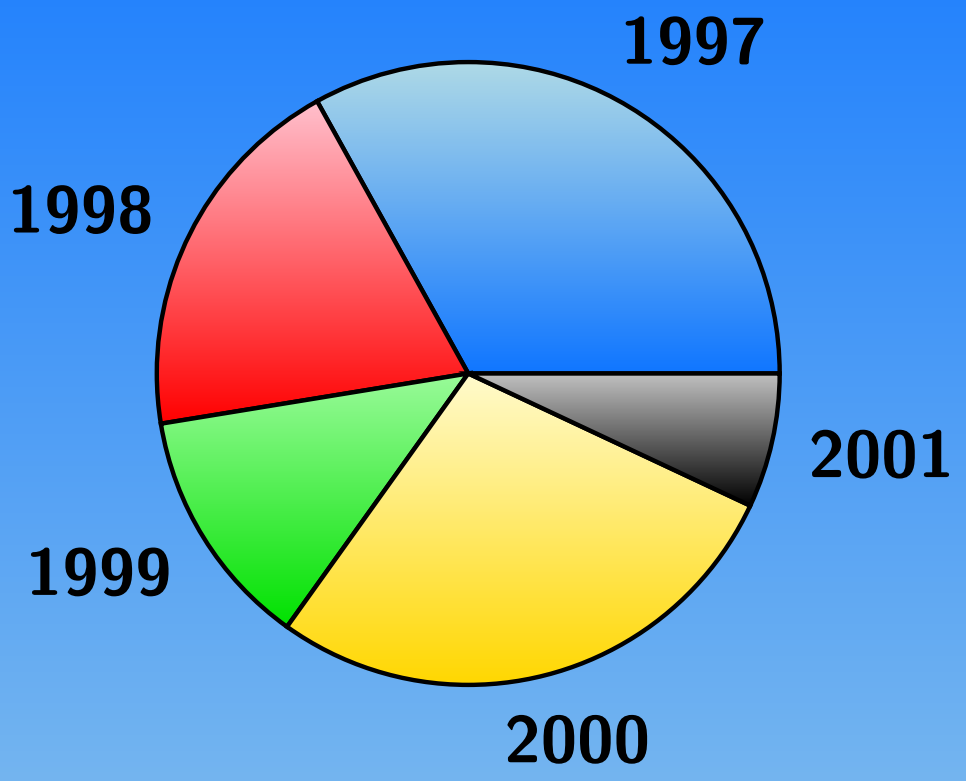


Figure 1: Results of the last five years



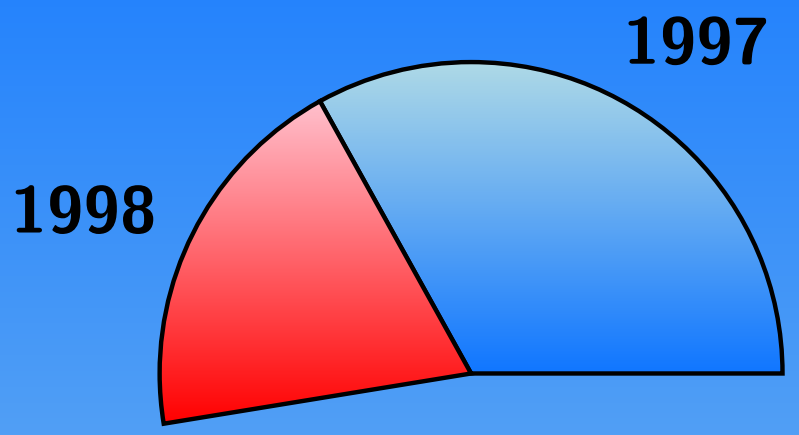


Figure 1: Results of the last five years

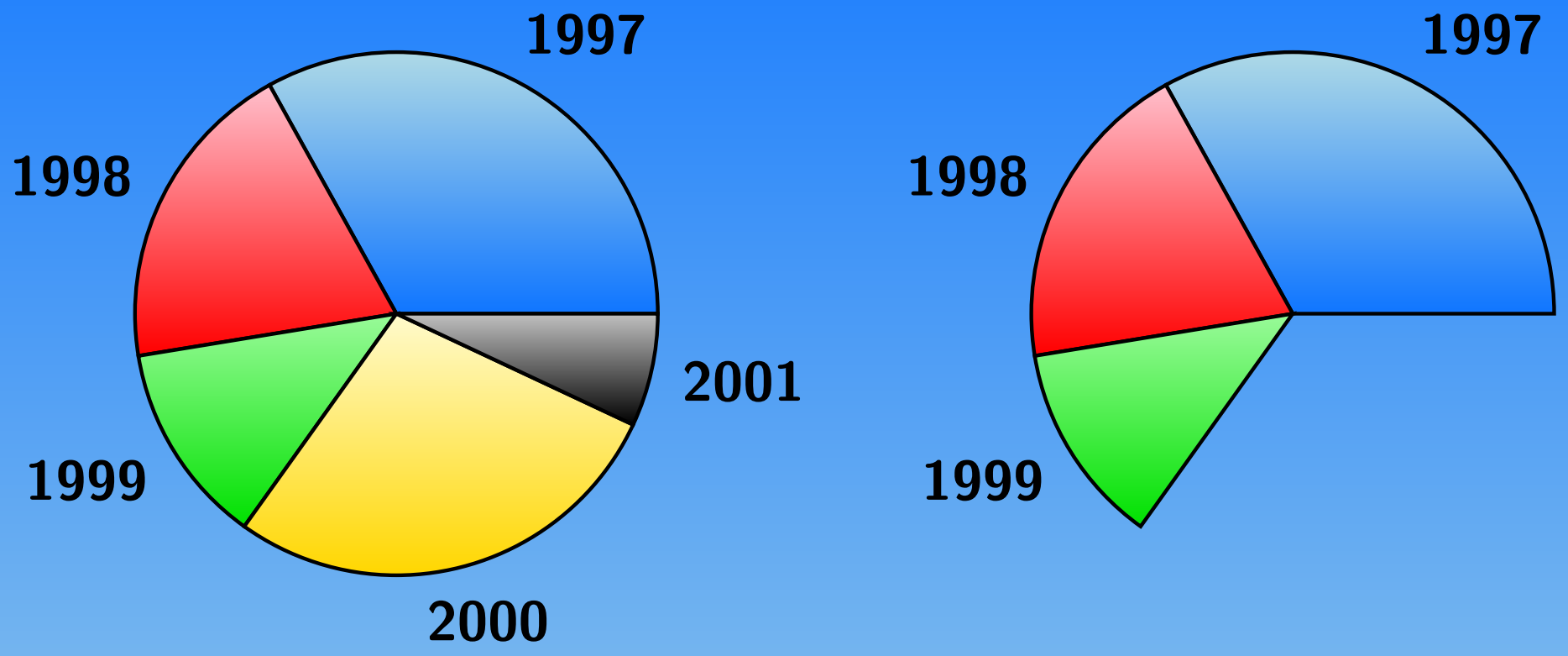


Figure 1: Results of the last five years

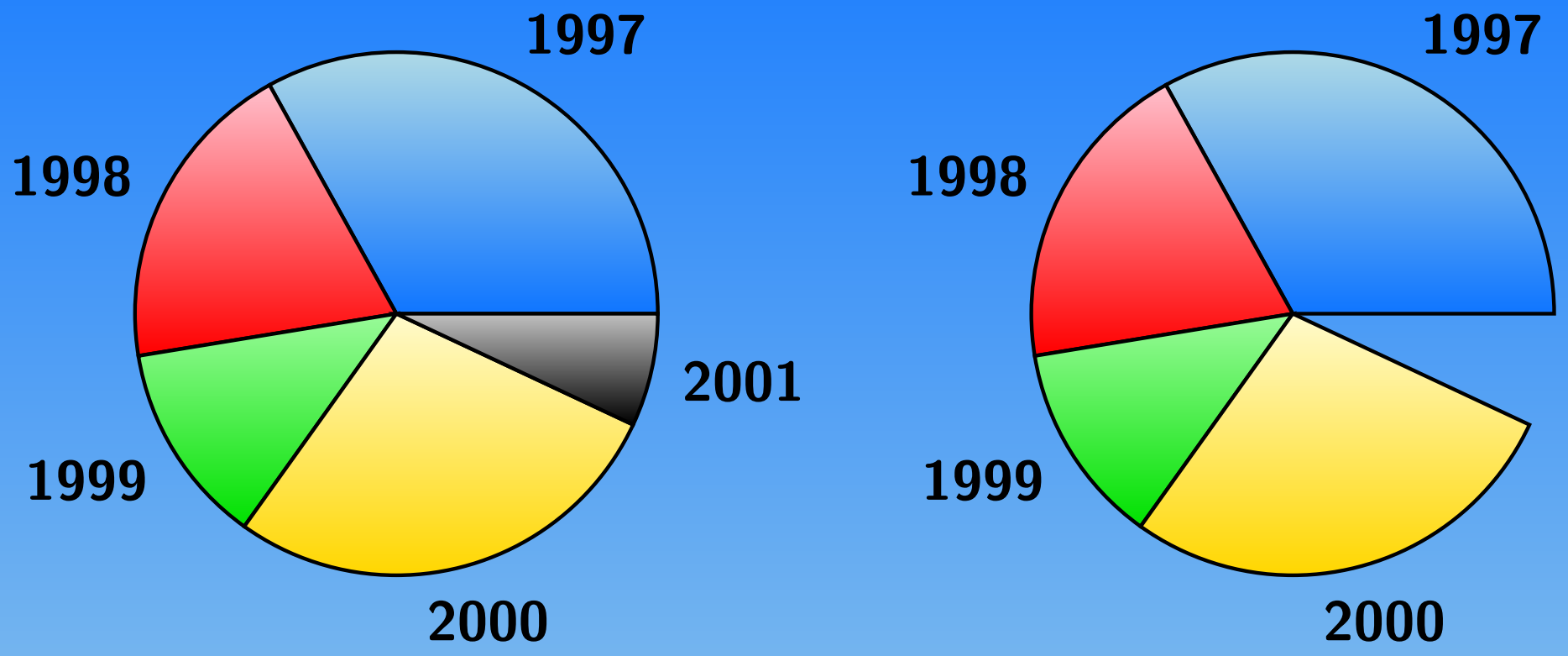


Figure 1: Results of the last five years

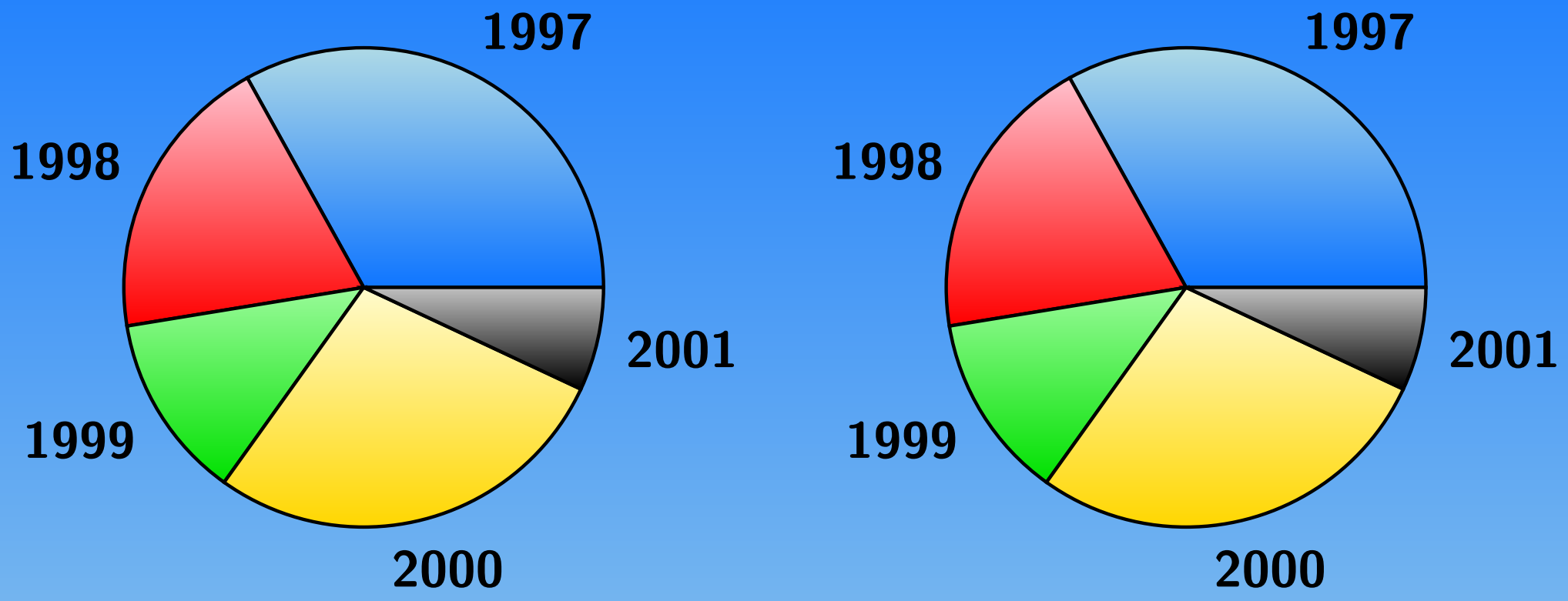


Figure 1: Results of the last five years

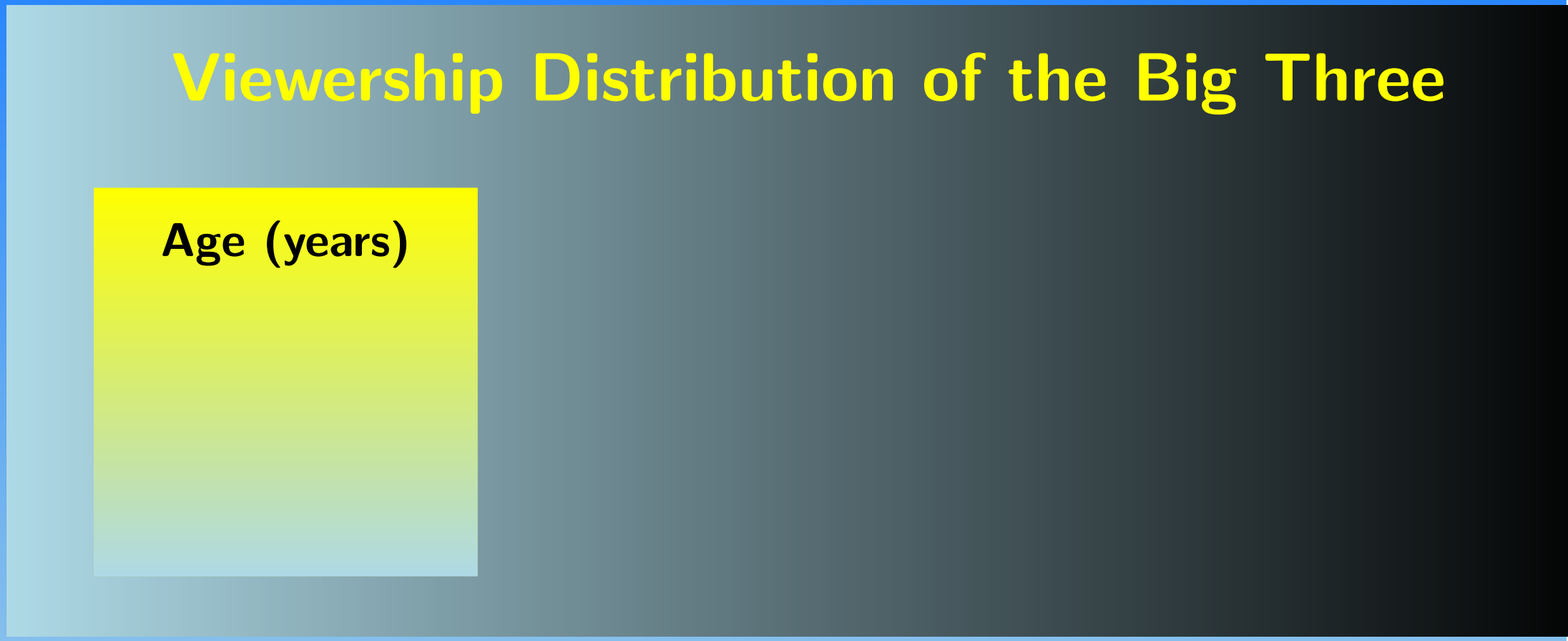


Figure 2: Main American TV channels

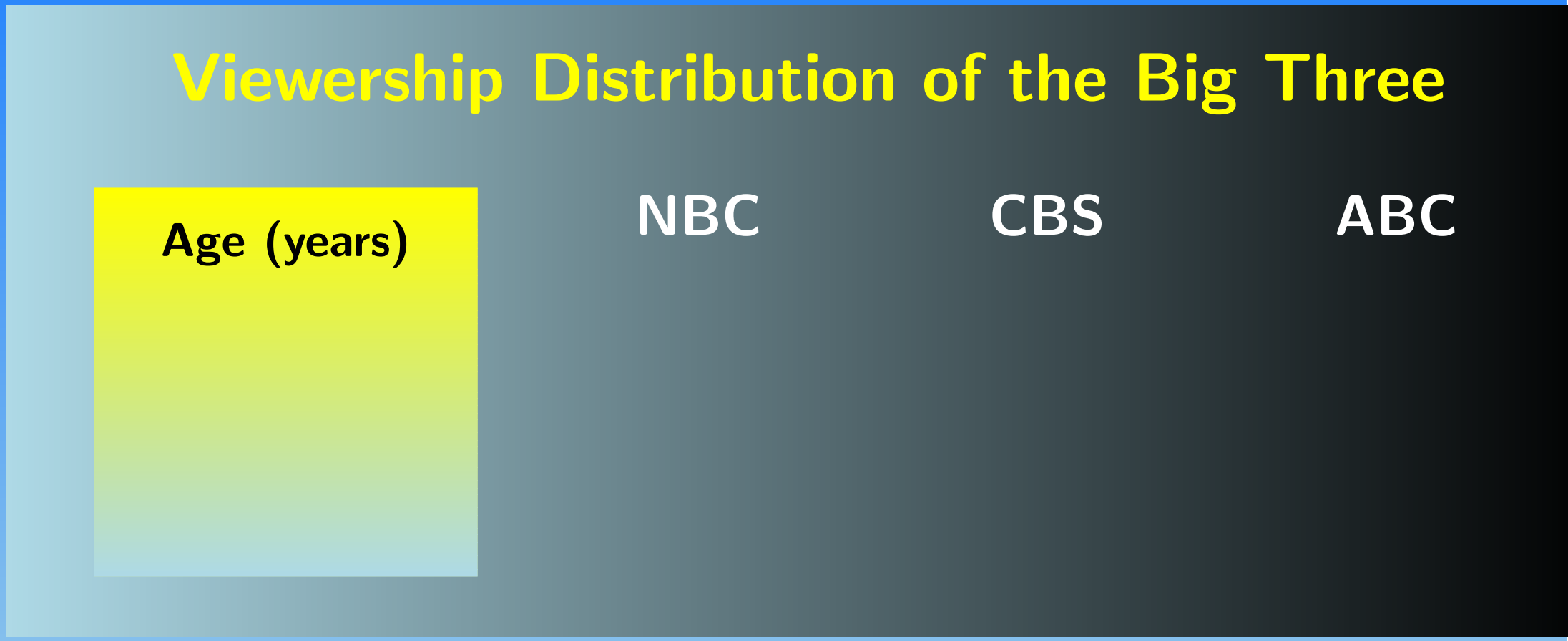


Figure 2: Main American TV channels

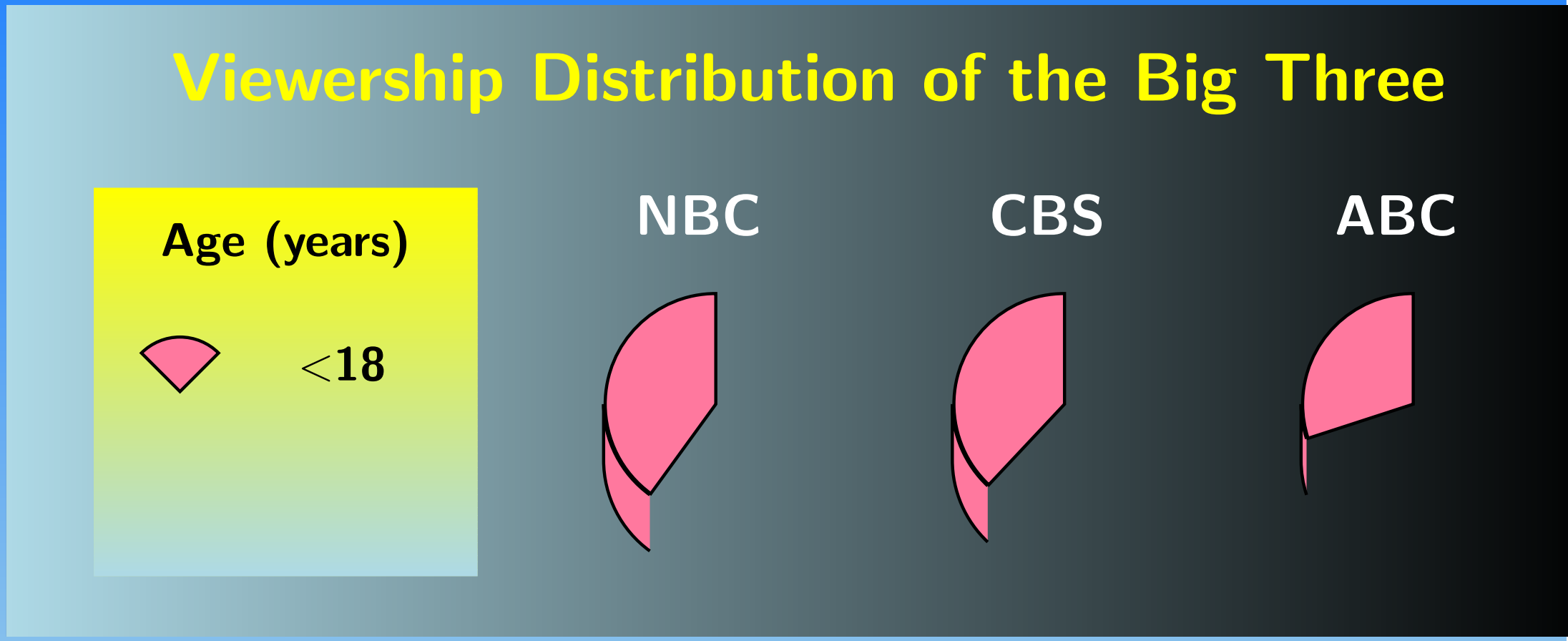


Figure 2: Main American TV channels

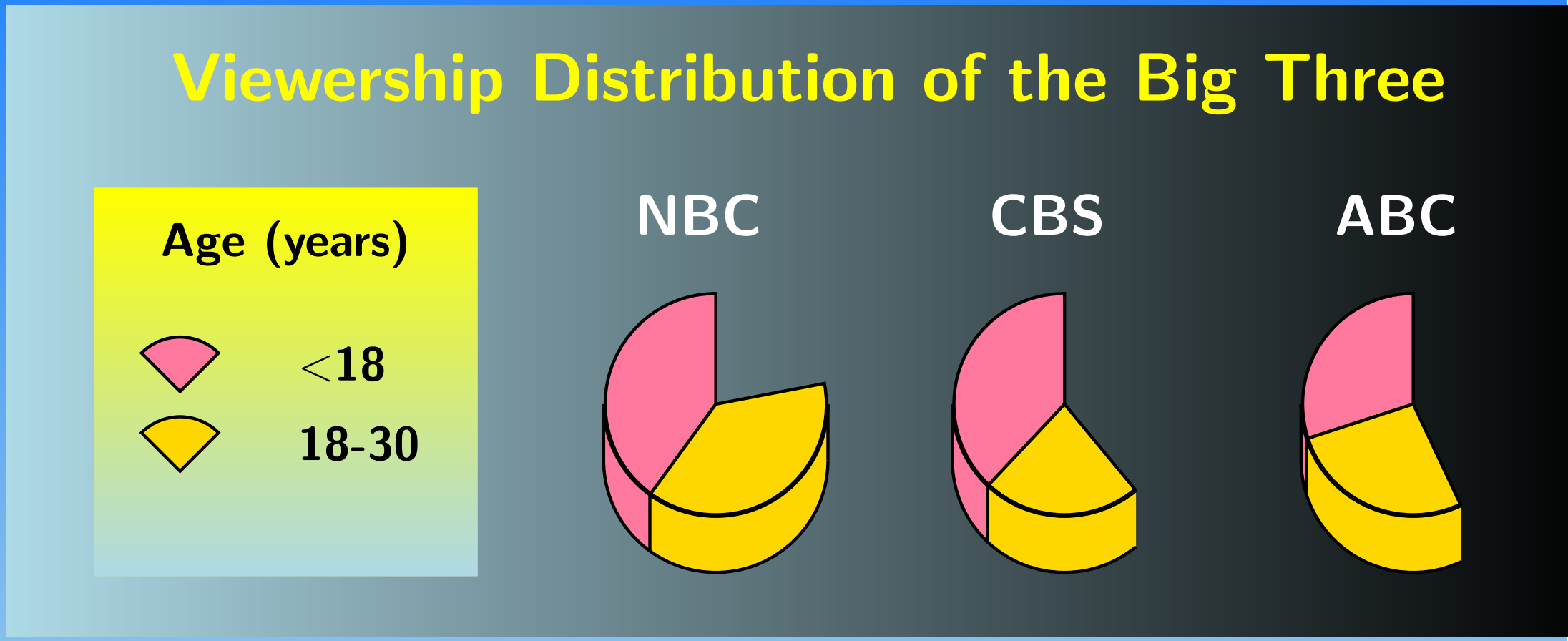


Figure 2: Main American TV channels

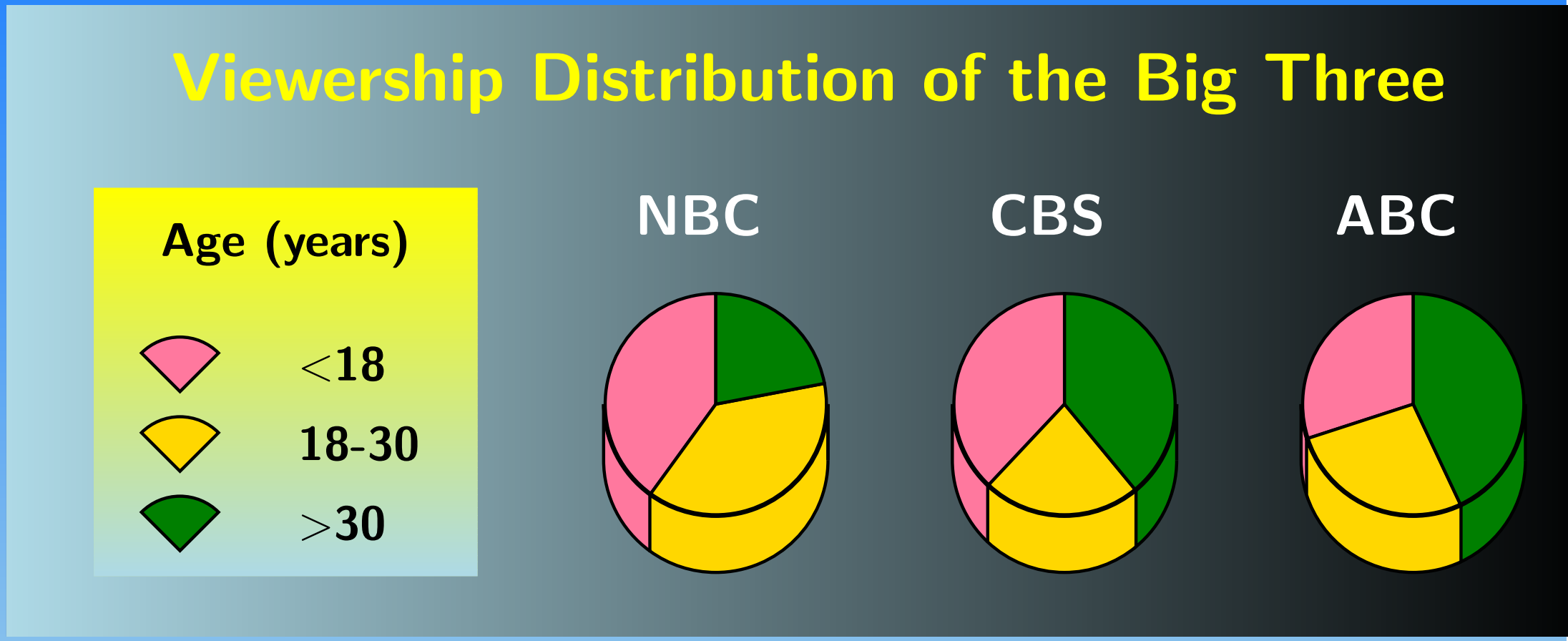


Figure 2: Main American TV channels

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi$$

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

= 2.44949

$$\Pi = \sqrt{6}$$

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{}$$

= 2.44949

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{1}$$

= 2.44949

1 $\Longrightarrow$ 2.44949

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{1 + \frac{1}{4}}$$

= 2.44949

1 $\Longrightarrow$ 2.44949

1.25 $\Longrightarrow$ 2.73861

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{1 + \frac{1}{4} + \frac{1}{9}}$$

= 2.44949

1 $\Longrightarrow$ *2.44949*

1.25 $\Longrightarrow$ *2.73861*

1.36111 $\Longrightarrow$ *2.85774*

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16}}$$

= 2.44949

1 $\Longrightarrow$ *2.44949*

1.25 $\Longrightarrow$ *2.73861*

1.36111 $\Longrightarrow$ *2.85774*

1.42361 $\Longrightarrow$ *2.92261*

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots}$$

= 2.44949

1 $\Longrightarrow$ *2.44949*

1.25 $\Longrightarrow$ *2.73861*

1.36111 $\Longrightarrow$ *2.85774*

1.42361 $\Longrightarrow$ *2.92261*

End of slide

Demonstration of overlays (ii)

2 – Equations with (cumulative) annotations

A formula for Π from Leonhard Euler

$$\Pi = \sqrt{6} \times \sqrt{1 + \frac{1}{4} + \frac{1}{9} + \frac{1}{16} + \cdots}$$

= 2.44949

1 $\Longrightarrow$ *2.44949*

1.25 $\Longrightarrow$ *2.73861*

1.36111 $\Longrightarrow$ *2.85774*

1.42361 $\Longrightarrow$ *2.92261*

$$= \left(6 \sum_{n=1}^{\infty} \frac{1}{n^2}\right)^{\frac{1}{2}}$$

End of slide

Demonstration of overlays (ii)

3 – Listing with (progressive) annotations

- From a manual to introduce to parallel programming with the MPI library
- First with overlays but without annotations, just using the features of the ‘fancyvrb’ package

```
program WhoAmI
  implicit none
  include ’mpif.h’
  integer :: nb_procs,rank,code




  print *,’I am process ’,rank,’ among ’,nb_procs


end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

3 – Listing with (progressive) annotations

- From a manual to introduce to parallel programming with the MPI library
- First with overlays but without annotations, just using the features of the ‘fancyvrb’ package

```
program WhoAmI
  implicit none
  include ’mpif.h’
  integer :: nb_procs,rank,code

  call MPI_INIT(code)




  print *,’I am process ’,rank,’ among ’,nb_procs


end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

3 – Listing with (progressive) annotations

- From a manual to introduce to parallel programming with the MPI library
- First with overlays but without annotations, just using the features of the ‘fancyvrb’ package

```
program WhoAmI
  implicit none
  include ’mpif.h’
  integer :: nb_procs,rank,code

  call MPI_INIT(code)




  print *,’I am process ’,rank,’ among ’,nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

3 – Listing with (progressive) annotations

- From a manual to introduce to parallel programming with the MPI library
- First with overlays but without annotations, just using the features of the ‘fancyvrb’ package

```
program WhoAmI
  implicit none
  include ’mpif.h’
  integer :: nb_procs,rank,code

  call MPI_INIT(code)

  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)


  print *,’I am process ’,rank,’ among ’,nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

3 – Listing with (progressive) annotations

- ☞ From a manual to introduce to parallel programming with the MPI library
- ☞ First with overlays but without annotations, just using the features of the ‘fancyvrb’ package

```
program WhoAmI
  implicit none
  include ’mpif.h’
  integer :: nb_procs,rank,code

  call MPI_INIT(code)

  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
  call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)

  print *,’I am process ’,rank,’ among ’,nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

- Then the same code, but using both the features of the ‘fancyvrb’ and ‘listings’ packages, with an automatic *pretty printing* of the code, after definition of a new language to emphasize the MPI-1 constants and subroutines (see the file `lstlang0.sty`)
- We could also use the ‘listings’ package alone

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code




  print *,'I am process ',rank,' among ',nb_procs


end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

- Then the same code, but using both the features of the ‘fancyvrb’ and ‘listings’ packages, with an automatic *pretty printing* of the code, after definition of a new language to emphasize the MPI-1 constants and subroutines (see the file `lstlang0.sty`)
- We could also use the ‘listings’ package alone

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code

  call MPI_INIT(code)




  print *,'I am process ',rank,' among ',nb_procs


end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

- Then the same code, but using both the features of the ‘fancyvrb’ and ‘listings’ packages, with an automatic *pretty printing* of the code, after definition of a new language to emphasize the MPI-1 constants and subroutines (see the file `lstlang0.sty`)
- We could also use the ‘listings’ package alone

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code

  call MPI_INIT(code)




  print *,'I am process ',rank,' among ',nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

- Then the same code, but using both the features of the ‘fancyvrb’ and ‘listings’ packages, with an automatic *pretty printing* of the code, after definition of a new language to emphasize the MPI-1 constants and subroutines (see the file `lstlang0.sty`)
- We could also use the ‘listings’ package alone

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code

  call MPI_INIT(code)

  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)


  print *,'I am process ',rank,' among ',nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

- Then the same code, but using both the features of the ‘fancyvrb’ and ‘listings’ packages, with an automatic *pretty printing* of the code, after definition of a new language to emphasize the MPI-1 constants and subroutines (see the file `lstlang0.sty`)
- We could also use the ‘listings’ package alone

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code

  call MPI_INIT(code)

  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
  call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)

  print *,'I am process ',rank,' among ',nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

☞ And now always the same code, but adding external annotations, using PSTricks nodes. This time, all annotations are shown together, without using overlays.

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9    call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)
10
11   print *,'I am process ',rank,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

Number of processes for the current execution

Rank of the process among all of them

Exit of MPI environment

End of slide

Demonstration of overlays (ii)

☞ Then finally the same code again, with the same annotations, but shown in a progressive way, using overlays

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code

  call MPI_INIT(code)

  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
  call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)

  print *,'I am process ',rank,' among ',nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

☞ Then finally the same code again, with the same annotations, but shown in a progressive way, using overlays

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9    call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)
10
11   print *,'I am process ',rank,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

End of slide

Demonstration of overlays (ii)

☞ Then finally the same code again, with the same annotations, but shown in a progressive way, using overlays

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9    call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)
10
11   print *,'I am process ',rank,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Number of processes for the current execution

End of slide

Demonstration of overlays (ii)

☞ Then finally the same code again, with the same annotations, but shown in a progressive way, using overlays

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9    call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)
10
11   print *,'I am process ',rank,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Rank of the process among all of them

End of slide

Demonstration of overlays (ii)

☞ Then finally the same code again, with the same annotations, but shown in a progressive way, using overlays

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code

  call MPI_INIT(code)

  call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
  call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)

  print *,'I am process ',rank,' among ',nb_procs

  call MPI_FINALIZE(code)
end program WhoAmI
```

Exit of MPI environment

End of slide

4 – (Cumulative) listing with (cumulative) annotations (I)

```
program WhoAmI
  implicit none
  include 'mpif.h'
  integer :: nb_procs,rank,code




  print *,'I am process ',     ,' among ',


end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8
9
10
11   print *,'I am process ',     ,' among ',
12
13
14 end program WhoAmI
```

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8
9
10
11   print *,'I am process ',     ,' among ',
12
13
14 end program WhoAmI
```

Initialization of MPI environment

End of slide

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8
9
10
11   print *,'I am process ',     ,' among ',
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8
9
10
11   print *,'I am process ',     ,' among ',
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

Exit of MPI environment

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9
10
11   print *,'I am process ',     ,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

Exit of MPI environment

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9
10
11   print *,'I am process ',     ,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

Number of processes for the current execution

Exit of MPI environment

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9    call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)
10
11   print *,'I am process ',rank,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

Number of processes for the current execution

Exit of MPI environment

End of slide

Demonstration of overlays (ii)

4 – (Cumulative) listing with (cumulative) annotations (I)

```
1  program WhoAmI
2    implicit none
3    include 'mpif.h'
4    integer :: nb_procs,rank,code
5
6    call MPI_INIT(code)
7
8    call MPI_COMM_SIZE(MPI_COMM_WORLD,nb_procs,code)
9    call MPI_COMM_RANK(MPI_COMM_WORLD,rank,code)
10
11   print *,'I am process ',rank,' among ',nb_procs
12
13   call MPI_FINALIZE(code)
14 end program WhoAmI
```

Initialization of MPI environment

Number of processes for the current execution

Rank of the process among all of them

Exit of MPI environment

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double    alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double    alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

File of CORBA required headers

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double     alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

File of CORBA required headers

File of headers relative to the skeleton generated from the IDL interface by the IDL compiler

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double    alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

File of CORBA required headers

File of headers relative to the skeleton generated from the IDL interface by the IDL compiler

The class `ClassMatrix` must now be known inside the CORBA POA

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double     alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

File of CORBA required headers

File of headers relative to the skeleton generated from the IDL interface by the IDL compiler

The class `ClassMatrix` must now be known inside the CORBA POA

Constructor

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double     alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

File of CORBA required headers

File of headers relative to the skeleton generated from the IDL interface by the IDL compiler

The class `ClassMatrix` must now be known inside the CORBA POA

Constructor

Destructor

End of slide

Demonstration of overlays (ii)

5 – Listing with (cumulative) annotations (II)

☞ From a manual to introduce to distributed programming with CORBA

```
1  #include <iostream.h>
2  #include <fstream.h>
3
4  #include <OB/CORBA.h>
5  #include <export_skel.h>
6
7  class ClassMatrix : virtual public POA_Exporte {
8
9   private :
10   TypeMatrix A;
11
12  public :
13   ClassMatrix(double init);
14   ~ClassMatrix();
15
16   virtual void MultiplyVector(CORBA::Double     alpha,
17                               TypeVector_slice *vector)
18     throw(CORBA::SystemException);
19 };
```

File of CORBA required headers

File of headers relative to the skeleton generated from the IDL interface by the IDL compiler

The class `ClassMatrix` must now be known inside the CORBA POA

Constructor

Destructor

Definition of a service to multiply a matrix by a scalar and a vector, with a management of the exceptions done by CORBA

End of slide

Demonstration of overlays (ii)

```
20 // Implementation of the methods
21
22 ClassMatrix::ClassMatrix(double cste) {
23   long long i, j;
24
25   for (i = 0; i < N; i++) {
26     for (j = 0; j < N; j++) {
27       A[i][j] = 0.0;}}
28   for (i = 0; i < N; i++) {
29       A[i][i] = cste;}
30 }
31
32 ClassMatrix::~ClassMatrix() {
33   cout << "Destruction of the object" << endl;
34 }
35
36 void ClassMatrix::MultiplyVector(CORBA::Double    alpha,
37                                   TypeVector_slice *vector)
38   throw(CORBA::SystemException) {
39
40   long long i, j;
41   TypeVector tmp;
42
43   for (i = 0; i < N; i++) {
44     tmp[i] = 0.0;
45     for (j = 0; j < N; j++) {
46       tmp[i] = tmp[i] + alpha * A[i][j] * vector[j];
47     }
48   }
49   for (i = 0; i < N; i++) vector[i] = tmp[i];
50 }
```

End of slide

Demonstration of overlays (ii)

```
20 // Implementation of the methods
21
22 ClassMatrix::ClassMatrix(double cste) {
23   long long i, j;
24
25   for (i = 0; i < N; i++) {
26     for (j = 0; j < N; j++) {
27       A[i][j] = 0.0;}}
28   for (i = 0; i < N; i++) {
29       A[i][i] = cste;}
30 }
31
32 ClassMatrix::~ClassMatrix() {
33   cout << "Destruction of the object" << endl;
34 }
35
36 void ClassMatrix::MultiplyVector(CORBA::Double    alpha,
37                                   TypeVector_slice *vector)
38   throw(CORBA::SystemException) {
39
40   long long i, j;
41   TypeVector tmp;
42
43   for (i = 0; i < N; i++) {
44     tmp[i] = 0.0;
45     for (j = 0; j < N; j++) {
46       tmp[i] = tmp[i] + alpha * A[i][j] * vector[j];
47     }
48   }
49   for (i = 0; i < N; i++) vector[i] = tmp[i];
50 }
```

Implementation of the constructor: initialization of the matrix to the identity matrix

End of slide

Demonstration of overlays (ii)

```
20 // Implementation of the methods
21
22 ClassMatrix::ClassMatrix(double cste) {
23   long long i, j;
24
25   for (i = 0; i < N; i++) {
26     for (j = 0; j < N; j++) {
27       A[i][j] = 0.0;}}
28   for (i = 0; i < N; i++) {
29       A[i][i] = cste;}
30 }
31
32 ClassMatrix::~ClassMatrix() {
33   cout << "Destruction of the object" << endl;
34 }
35
36 void ClassMatrix::MultiplyVector(CORBA::Double     alpha,
37                                   TypeVector_slice *vector)
38   throw(CORBA::SystemException) {
39
40   long long i, j;
41   TypeVector tmp;
42
43   for (i = 0; i < N; i++) {
44     tmp[i] = 0.0;
45     for (j = 0; j < N; j++) {
46       tmp[i] = tmp[i] + alpha * A[i][j] * vector[j];
47     }
48   }
49   for (i = 0; i < N; i++) vector[i] = tmp[i];
50 }
```

Implementation of the constructor: initialization of the matrix to the identity matrix

Implementation of the destructor: generally memory deallocation

End of slide

Demonstration of overlays (ii)

```
20 // Implementation of the methods
21
22 ClassMatrix::ClassMatrix(double cste) {
23   long long i, j;
24
25   for (i = 0; i < N; i++) {
26     for (j = 0; j < N; j++) {
27       A[i][j] = 0.0;}}
28   for (i = 0; i < N; i++) {
29       A[i][i] = cste;}
30 }
31
32 ClassMatrix::~ClassMatrix() {
33   cout << "Destruction of the object" << endl;
34 }
35
36 void ClassMatrix::MultiplyVector(CORBA::Double     alpha,
37                                   TypeVector_slice *vector)
38   throw(CORBA::SystemException) {
39
40   long long i, j;
41   TypeVector tmp;
42
43   for (i = 0; i < N; i++) {
44     tmp[i] = 0.0;
45     for (j = 0; j < N; j++) {
46       tmp[i] = tmp[i] + alpha * A[i][j] * vector[j];
47     }
48   }
49   for (i = 0; i < N; i++) vector[i] = tmp[i];
50 }
```

Implementation of the constructor: initialization of the matrix to the identity matrix

Implementation of the destructor: generally memory deallocation

Service to compute the product of a matrix (multiplied by a constant) with a vector, with a management of the exceptions done by CORBA

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

Declaration and initialization of the POA

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

Declaration and initialization of the POA

Creation of an object `Matrix`

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

Declaration and initialization of the POA

Creation of an object `Matrix`

Local characters string, used to store the generated reference

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

Declaration and initialization of the POA

Creation of an object `Matrix`

Local characters string, used to store the generated reference

Writing of this server reference in the file `reference`

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

Declaration and initialization of the POA

Creation of an object `Matrix`

Local characters string, used to store the generated reference

Writing of this server reference in the file `reference`

Activation of the ORB (which will "listen")

End of slide

Demonstration of overlays (ii)

```
51 // Main program
52
53 int main(int argc, char* argv[])
54 {
55   // Initialization of the CORBA ORB and POA
56   CORBA::ORB_var orb = CORBA::ORB_init(argc, argv);
57   CORBA::Object_var poaObj = orb -> resolve_initial_references("RootPOA");
58   PortableServer::POA_var RootPOA = PortableServer::POA::_narrow(poaObj);
59
60   ClassMatrix Matrix((double) (1.0));
61
62   // Writing of the "universal pointer" IOR in the file "reference"
63   CORBA::String_var str = orb -> object_to_string( Matrix._this() );
64   ofstream out("reference");
65   out << str << endl;
66   out.close();
67
68   RootPOA -> the_POAManager() -> activate();
69   orb -> run();
70
71   orb -> destroy();
72 }
```

Declaration and initialization of the ORB

Declaration and initialization of the POA

Creation of an object `Matrix`

Local characters string, used to store the generated reference

Writing of this server reference in the file `reference`

Activation of the ORB (which will "listen")

Destruction of the ORB (it will never occur here)

End of slide

2 – Communication ring

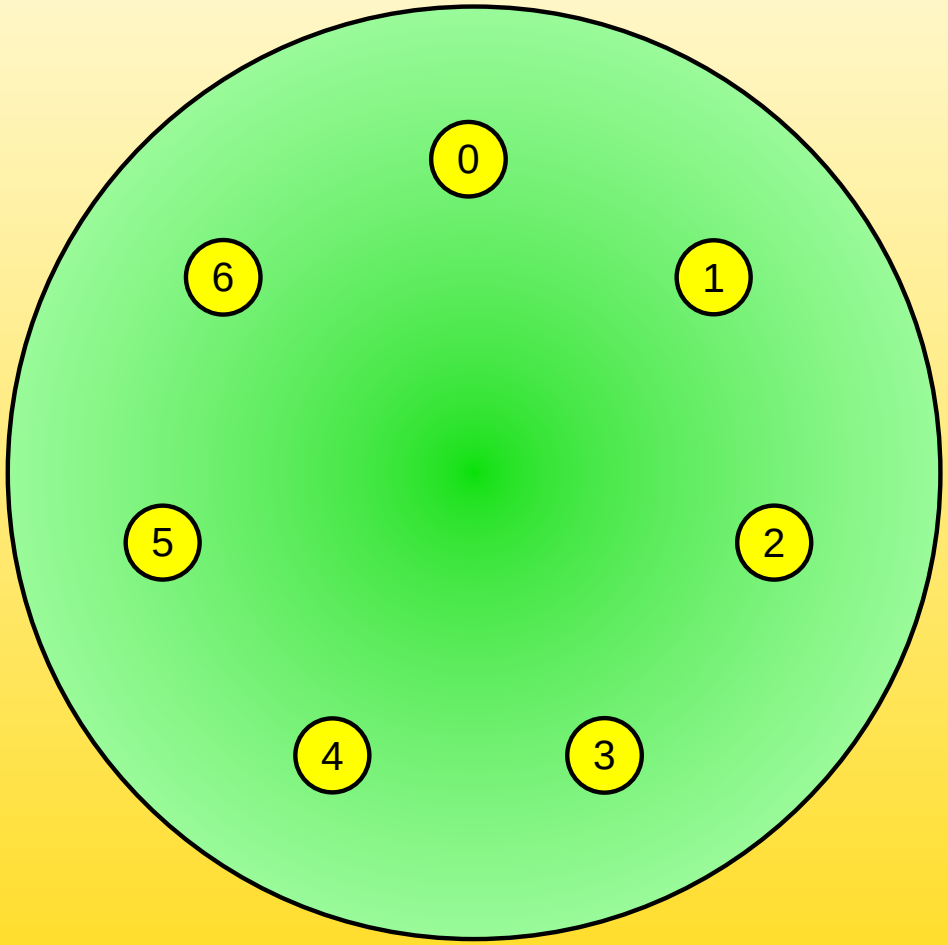


Figure 1: Communication ring

End of animation

2 – Communication ring

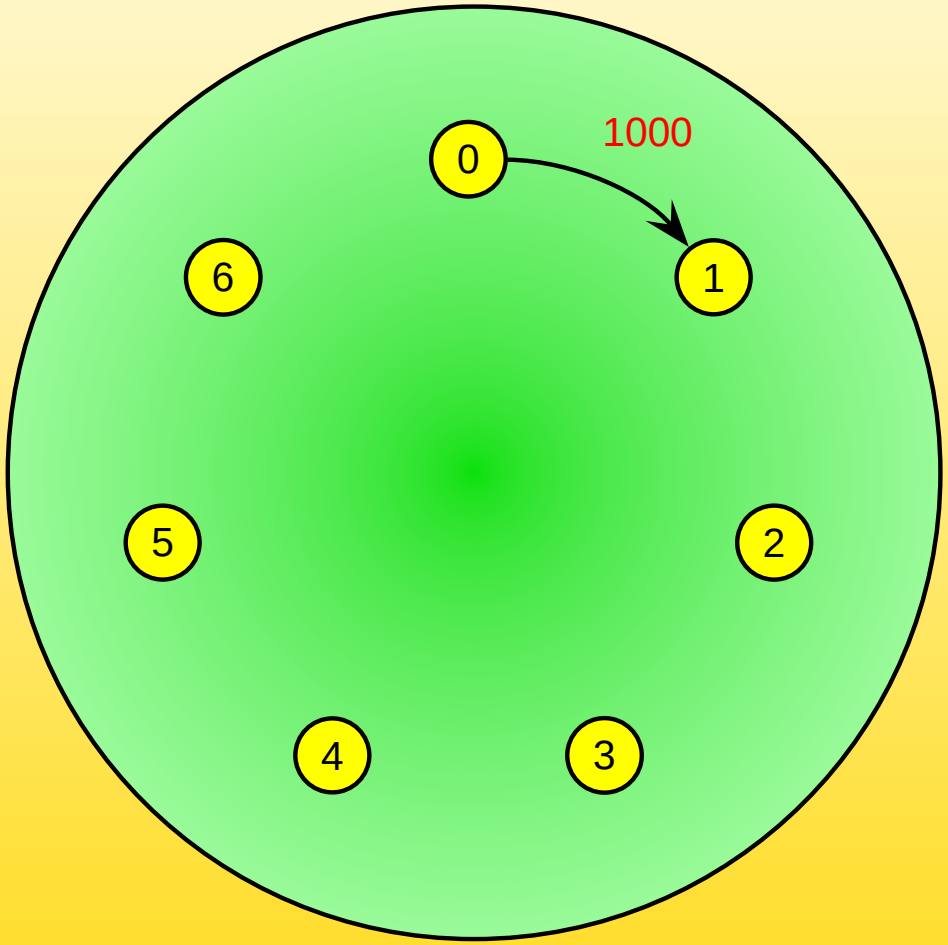


Figure 1: Communication ring

End of animation

2 – Communication ring

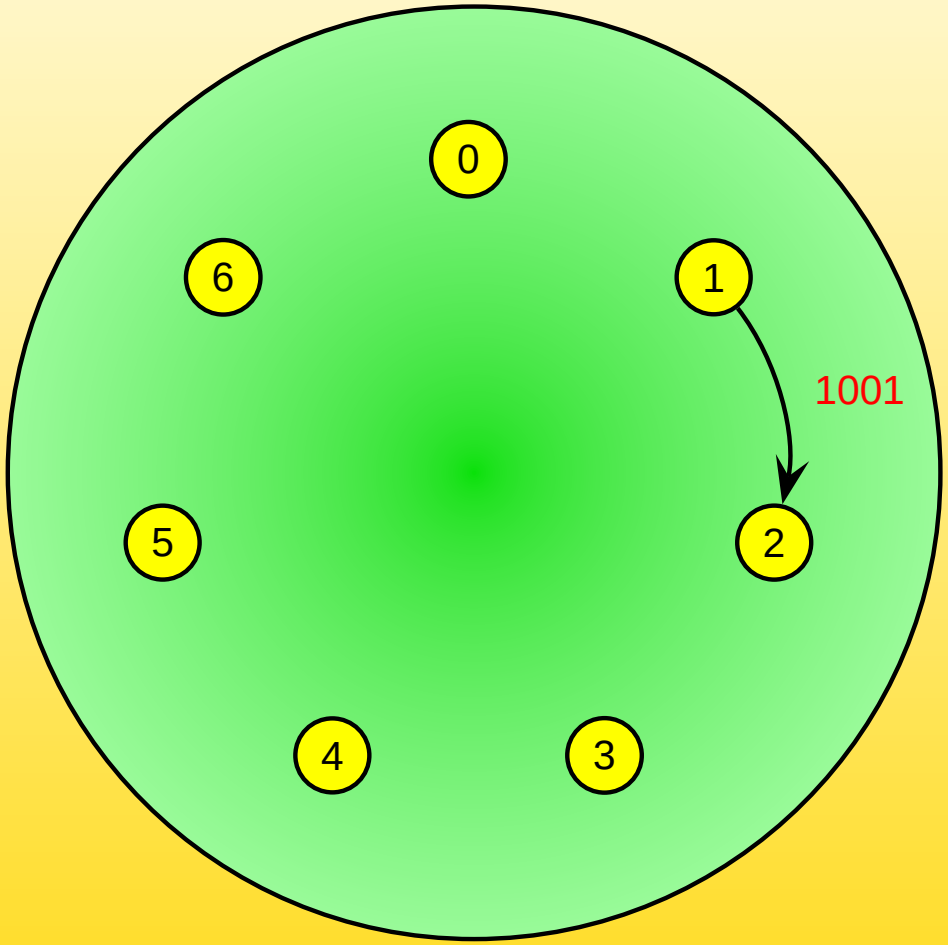


Figure 1: Communication ring

End of animation

2 – Communication ring

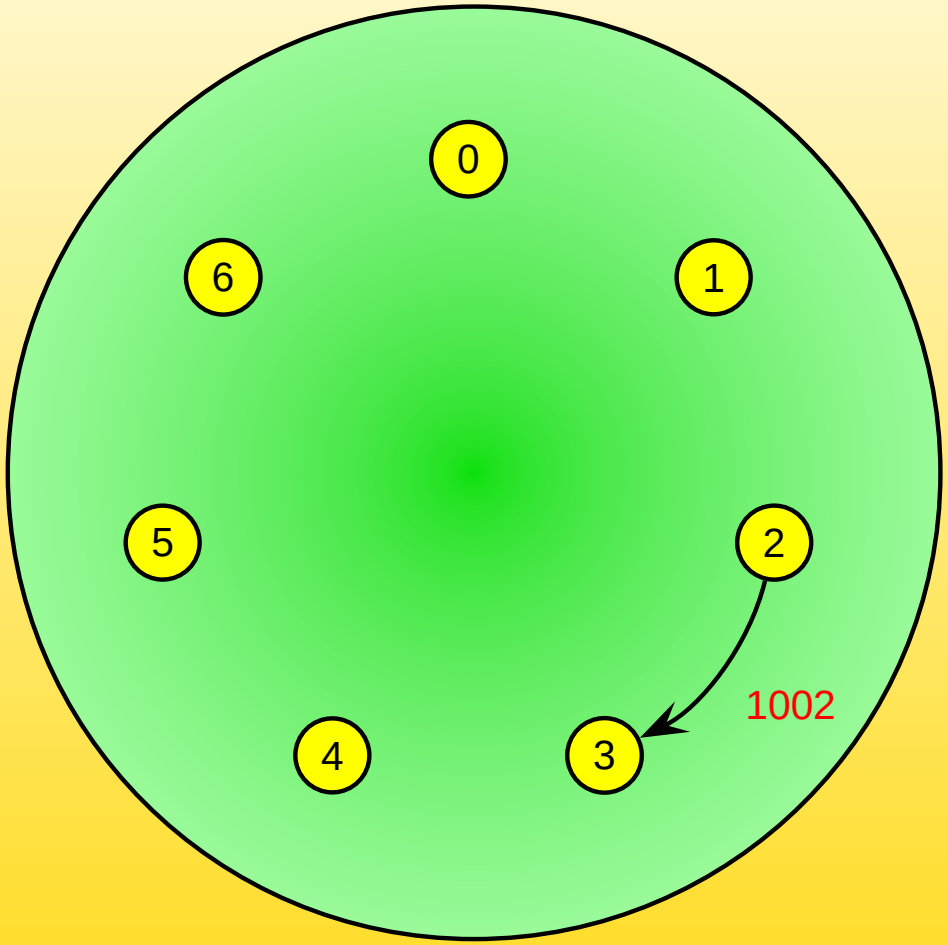


Figure 1: Communication ring

End of animation

2 – Communication ring

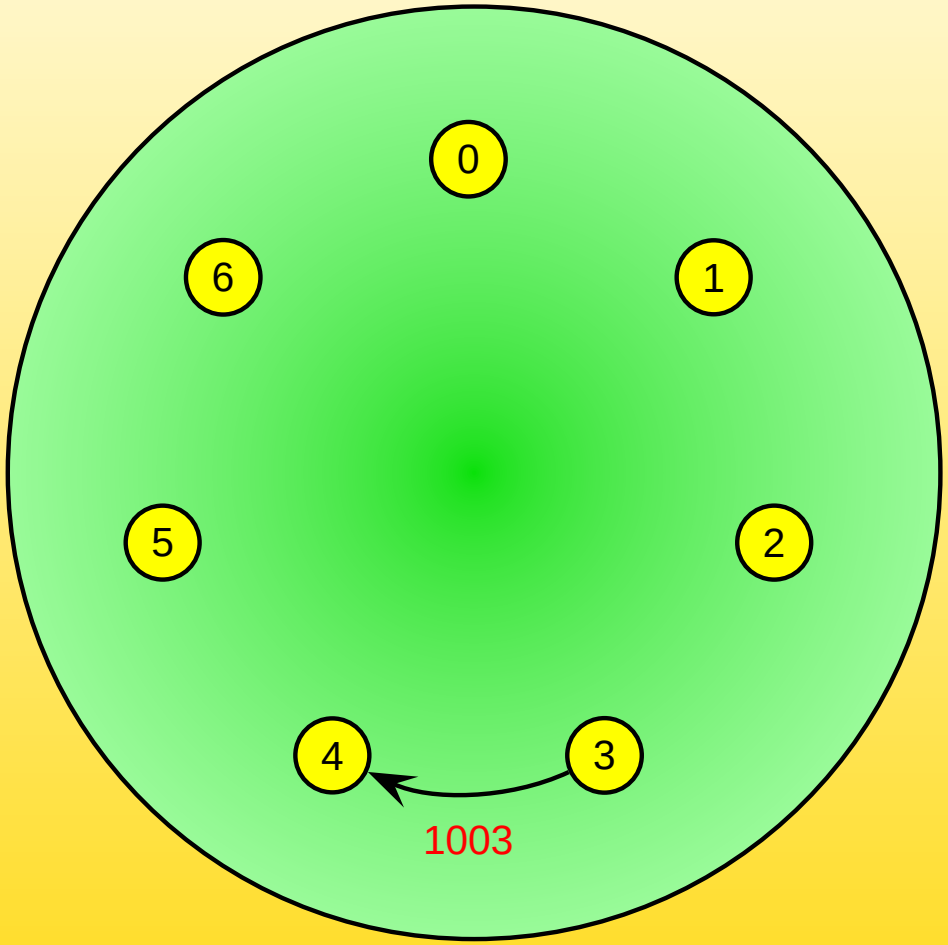


Figure 1: Communication ring

End of animation

2 – Communication ring

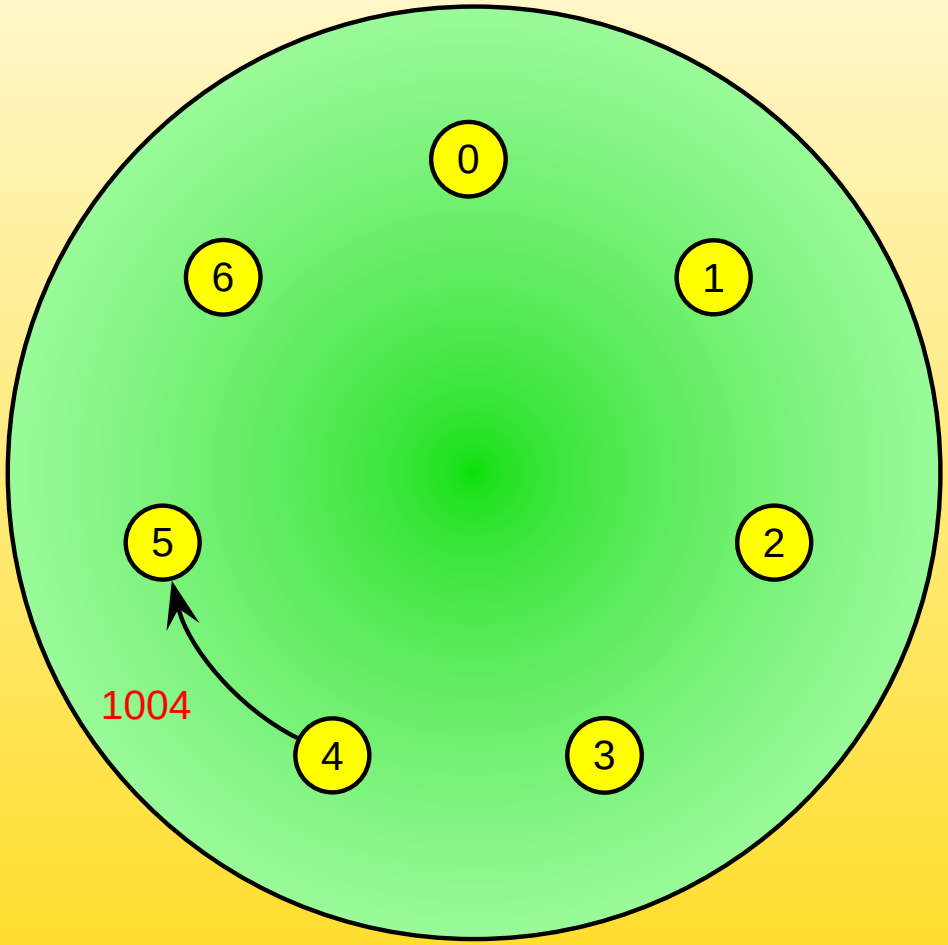


Figure 1: Communication ring

End of animation

2 – Communication ring

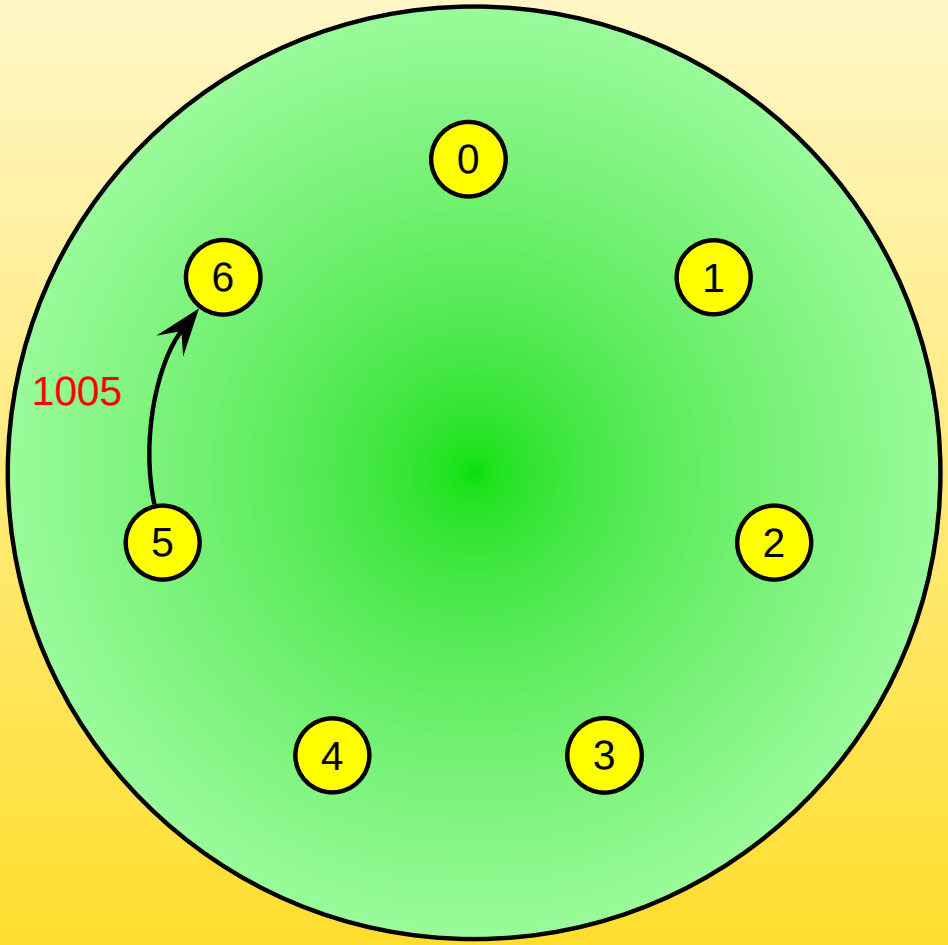


Figure 1: Communication ring

End of animation

2 – Communication ring

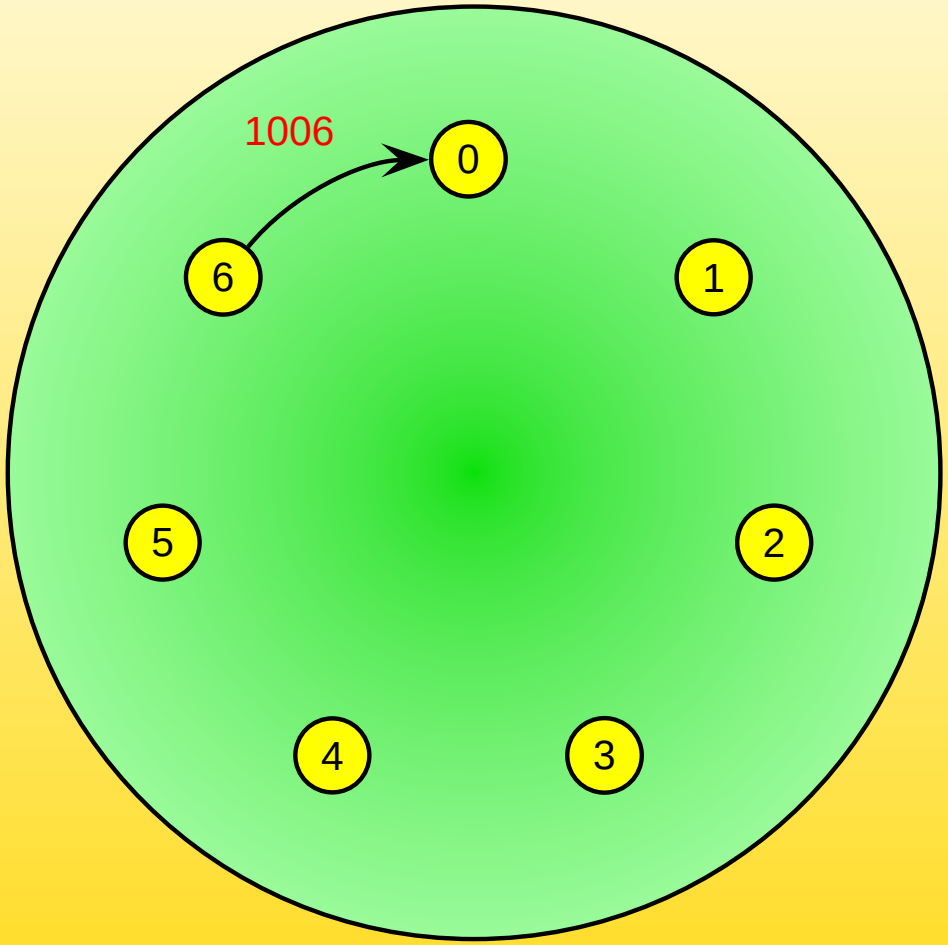


Figure 1: Communication ring

End of animation

3 – Commutative diagram

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

L

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$L \xleftarrow{\quad i_1 \quad} L_r$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$L \xleftarrow{\quad i_1 \quad} L_r \xrightarrow{\quad r \quad} R$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$\begin{tikzcd} L & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \\ L_m \arrow[u, "i_2"] & & \end{tikzcd}$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$\begin{tikzcd}
L & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \\
L_m \arrow[u, "i_2"] & K_{r,m} \arrow[l, "i_3"'] \arrow[u, "i_4"'] &
\end{tikzcd}$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
L & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \\
L_m \arrow[u, "i_2"] & K_{r,m} \arrow[l, "i_3"'] \arrow[r, "r"] \arrow[u, "i_4"] & R_{m^*} \arrow[u, "i_6"]
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
L & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \\
L_m \arrow[u, "i_2"] \arrow[d, "m"'] & K_{r,m} \arrow[l, "i_3"'] \arrow[r, "r"] \arrow[u, "i_4"] & R_{m^*} \arrow[u, "i_6"'] \\
G & &
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
L & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \\
L_m \arrow[u, "i_2"] \arrow[d, "m"'] & K_{r,m} \arrow[l, "i_3"'] \arrow[r, "r"] \arrow[u, "i_4"] \arrow[d, "m"] & R_{m^*} \arrow[u, "i_6"'] \\
G & G_{r^*} \arrow[l, "i_5"] &
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
L & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \\
L_m \arrow[u, "i_2"] \arrow[d, "m"'] & K_{r,m} \arrow[l, "i_3"'] \arrow[r, "r"] \arrow[u, "i_4"] \arrow[d, "m"] & R_{m^*} \arrow[u, "i_6"'] \arrow[d, "m^*"] \\
G & G_{r^*} \arrow[l, "i_5"] \arrow[r, "r^*"'] & H
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
 & \Sigma^L & & \\
L \arrow[ur, "\lambda^L"] & & L_r \arrow[ll, "i_1"'] \arrow[rr, "r"] & & R \\
L_m \arrow[u, "i_2"] \arrow[d, "m"'] & & K_{r,m} \arrow[ll, "i_3"'] \arrow[rr, "r"] \arrow[u, "i_4"] \arrow[d, "m"] & & R_{m^*} \arrow[u, "i_6"'] \arrow[d, "m^*"] \\
G & & G_{r^*} \arrow[ll, "i_5"] \arrow[rr, "r^*"'] & & H
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
 & \Sigma^L \arrow[rr, "\varphi^r"] & & \Sigma^R \\
L \arrow[ur, "\lambda^L"] & L_r \arrow[l, "i_1"'] \arrow[r, "r"] & R \arrow[ur, "\lambda^R"'] & \\
L_m \arrow[u, "i_2"] \arrow[d, "m"'] & K_{r,m} \arrow[l, "i_3"'] \arrow[r, "r"] \arrow[u, "i_4"] \arrow[d, "m"] & R_{m^*} \arrow[u, "i_6"'] \arrow[d, "m^*"] & \\
G & G_{r^*} \arrow[l, "i_5"] \arrow[r, "r^*"'] & H &
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
 & \Sigma^L \arrow[rr, "\varphi^r"] \arrow[dd, "\varphi^m"] & & \Sigma^R \\
L \arrow[ur, "\lambda^L"] \arrow[dd, "i_2"] & & L_r \arrow[ll, "i_1"'] \arrow[r, "r"] \arrow[dd, "i_4"] & R \arrow[ur, "\lambda^R"'] \arrow[dd, "i_6"] \\
 & \Sigma^G & & \\
L_m \arrow[ur, "i_3"] \arrow[dd, "m"] & & K_{r,m} \arrow[ll, "i_3"'] \arrow[r, "r"] \arrow[dd, "m"] & R_{m^*} \arrow[dd, "m^*"] \\
 & & & \\
G \arrow[ur, "\lambda^G"] & & G_{r^*} \arrow[ll, "i_5"] \arrow[r, "r^*"'] & H
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

3 – Commutative diagram

$$
\begin{tikzcd}
 & \Sigma^L \arrow[rr, "\varphi^r"] \arrow[dd, "\varphi^m"] & & \Sigma^R \arrow[dd, "\varphi^{m^*}"] \\
L \arrow[ur, "\lambda^L"] & & L_r \arrow[ll, "i_1"'] \arrow[rr, "r"] \arrow[uu, "i_4"] & & R \arrow[ur, "\lambda^R"] \arrow[uu, "i_6"] \\
L_m \arrow[uu, "i_2"] \arrow[dd, "m"'] & & K_{r,m} \arrow[ll, "i_3"'] \arrow[rr, "r"] \arrow[uu, "i_4"] \arrow[dd, "m"] & & R_{m^*} \arrow[uu, "i_6"] \arrow[dd, "m^*"] \\
 & \Sigma^G \arrow[rr, "\varphi^{r^*}"'] & & \Sigma^H \\
G \arrow[ur, "\lambda^G"] & & G_{r^*} \arrow[ll, "i_5"] \arrow[rr, "r^*"'] & & H \arrow[ur, dashed, "\lambda^H"']
\end{tikzcd}
$$

Figure 2: Commutative diagram

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter

2nd Quarter

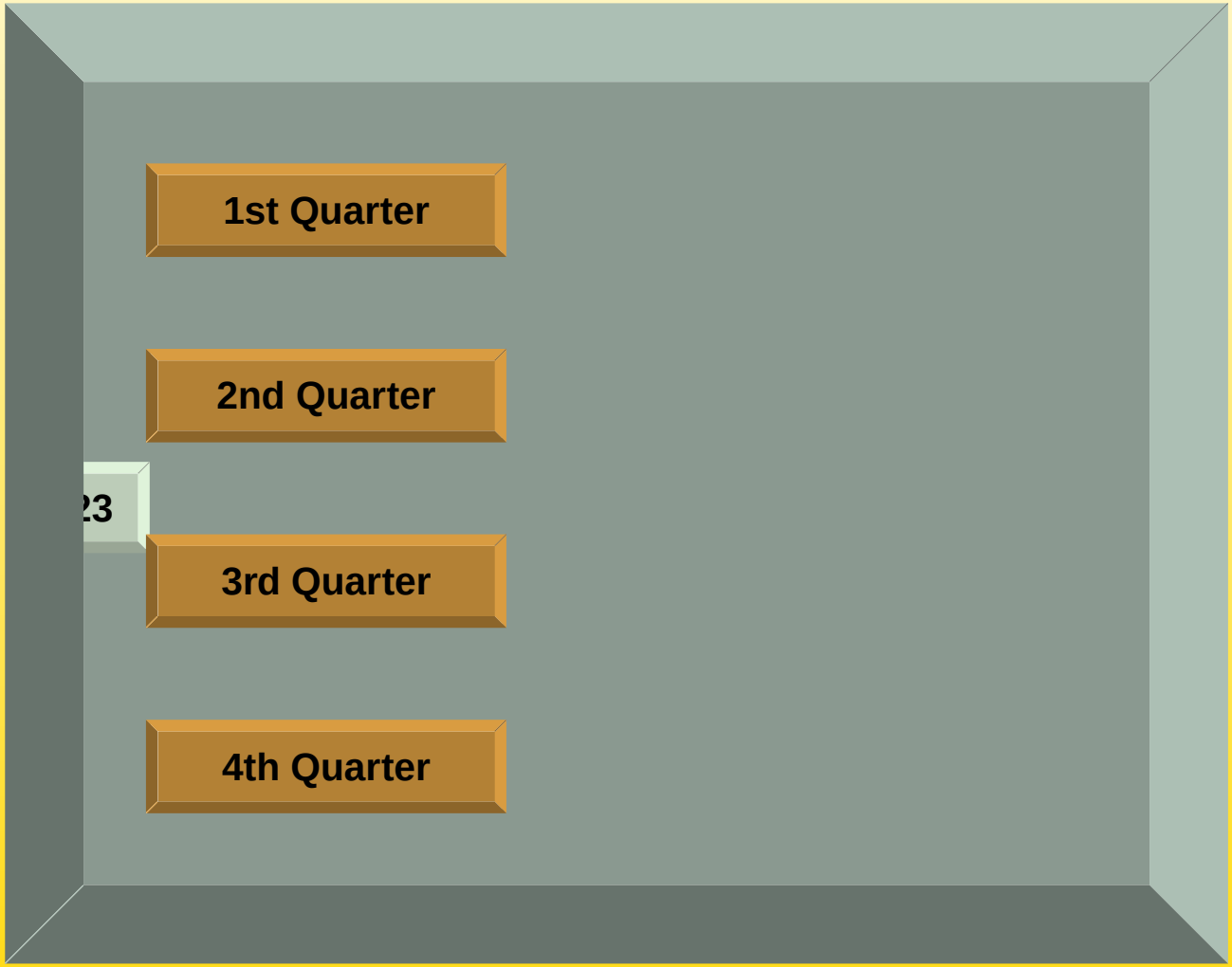
3rd Quarter

4th Quarter

End of animation

4 – Results of the year

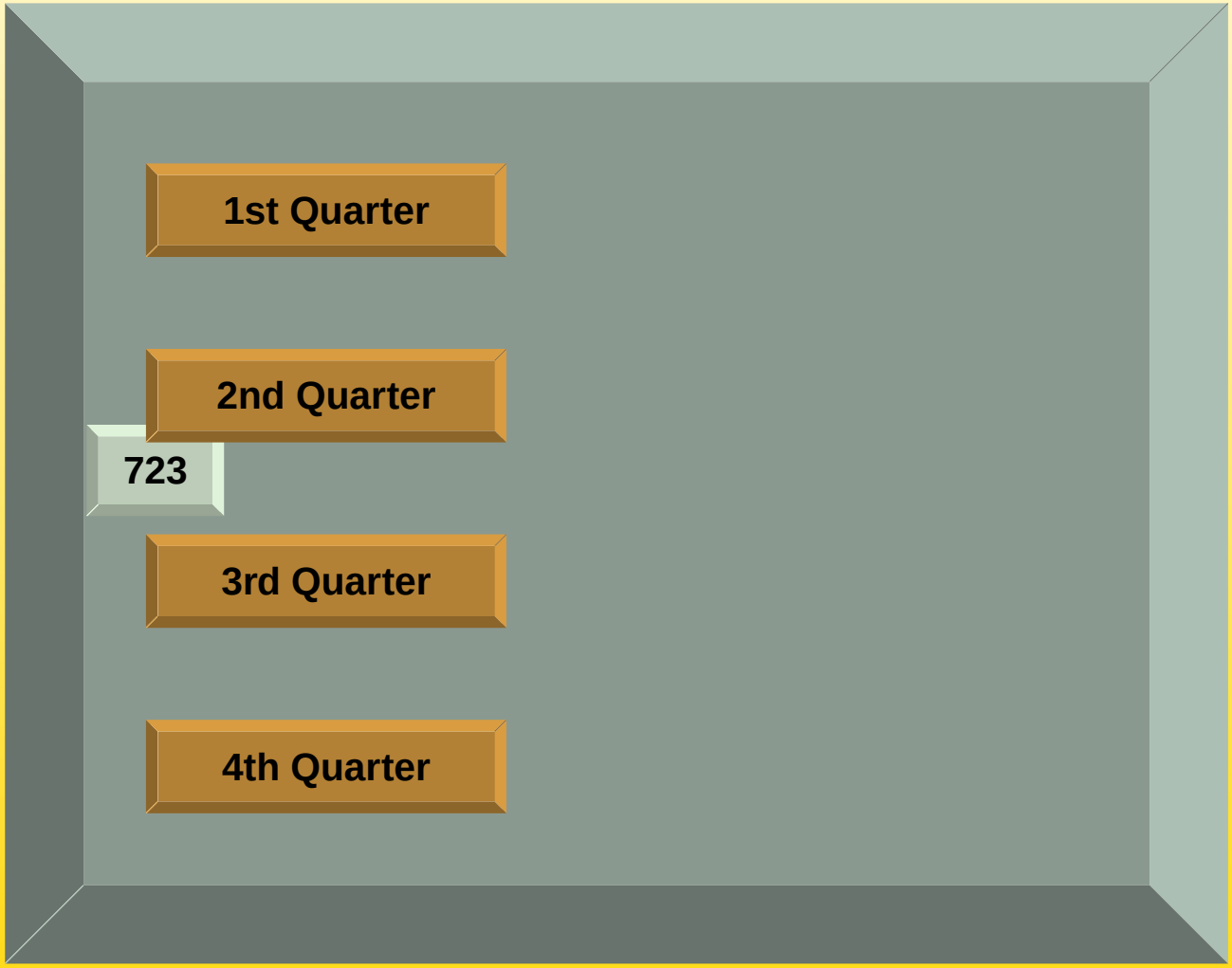
Table 1: Results of the year



End of animation

4 – Results of the year

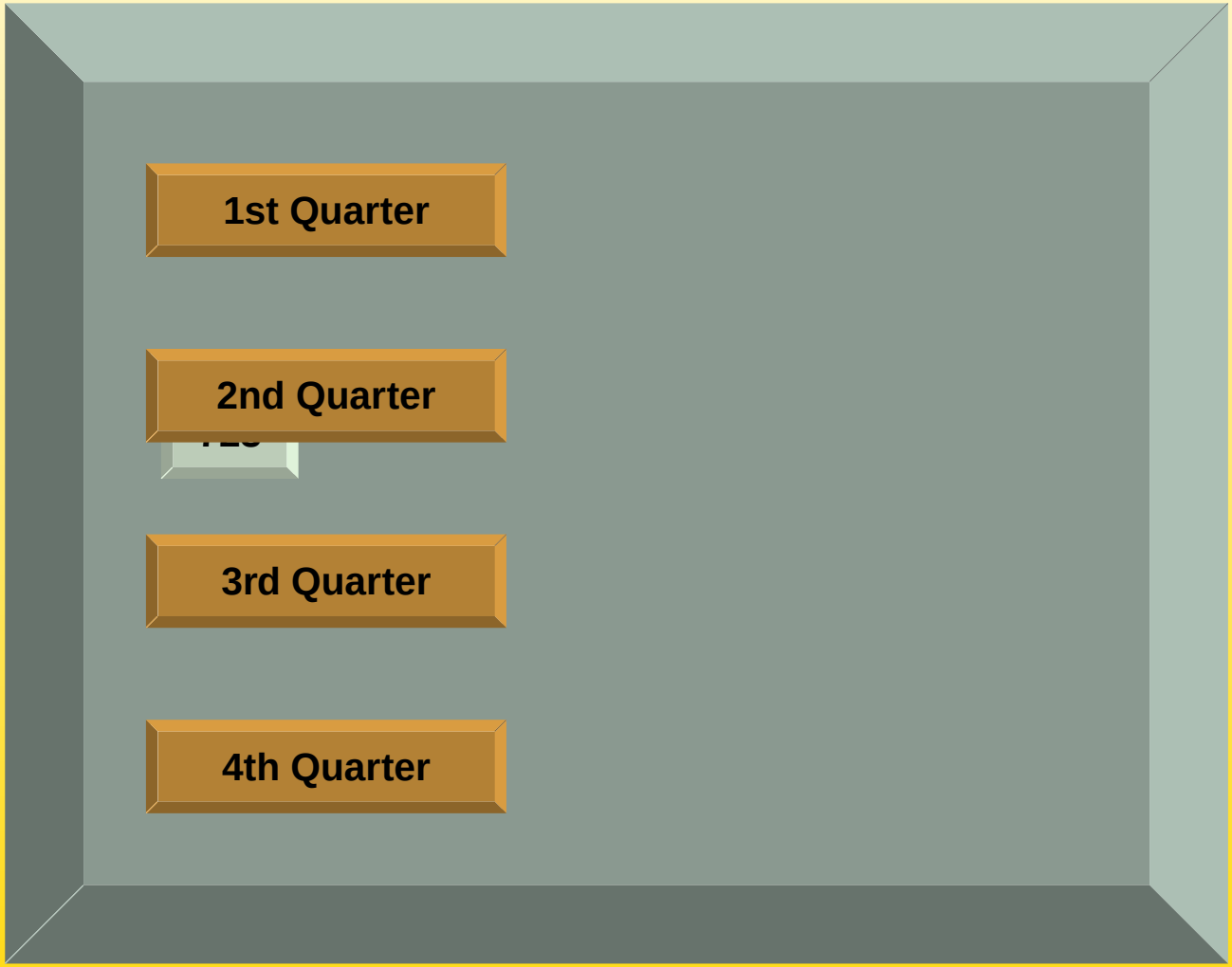
Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter

2nd Quarter

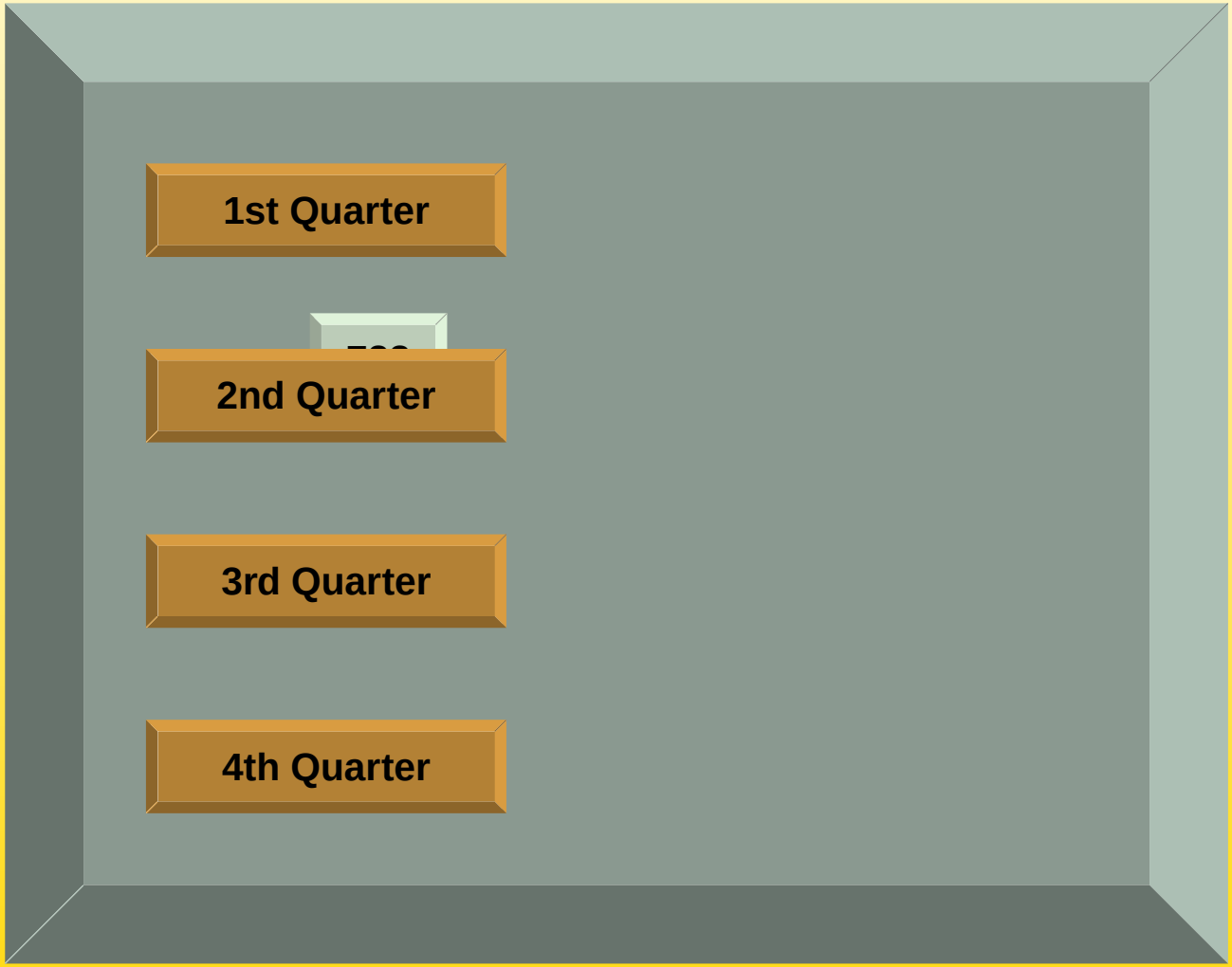
3rd Quarter

4th Quarter

End of animation

4 – Results of the year

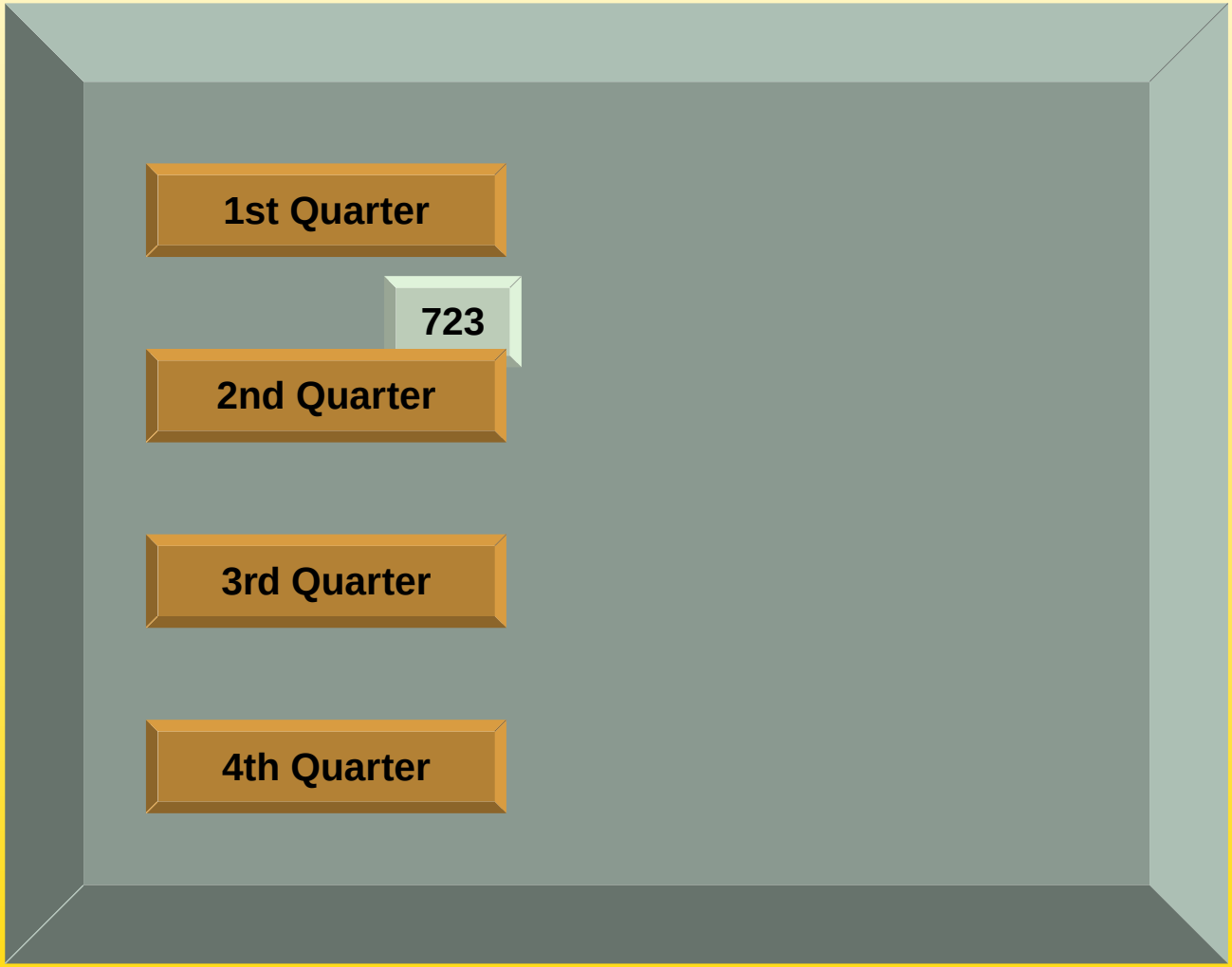
Table 1: Results of the year



End of animation

4 – Results of the year

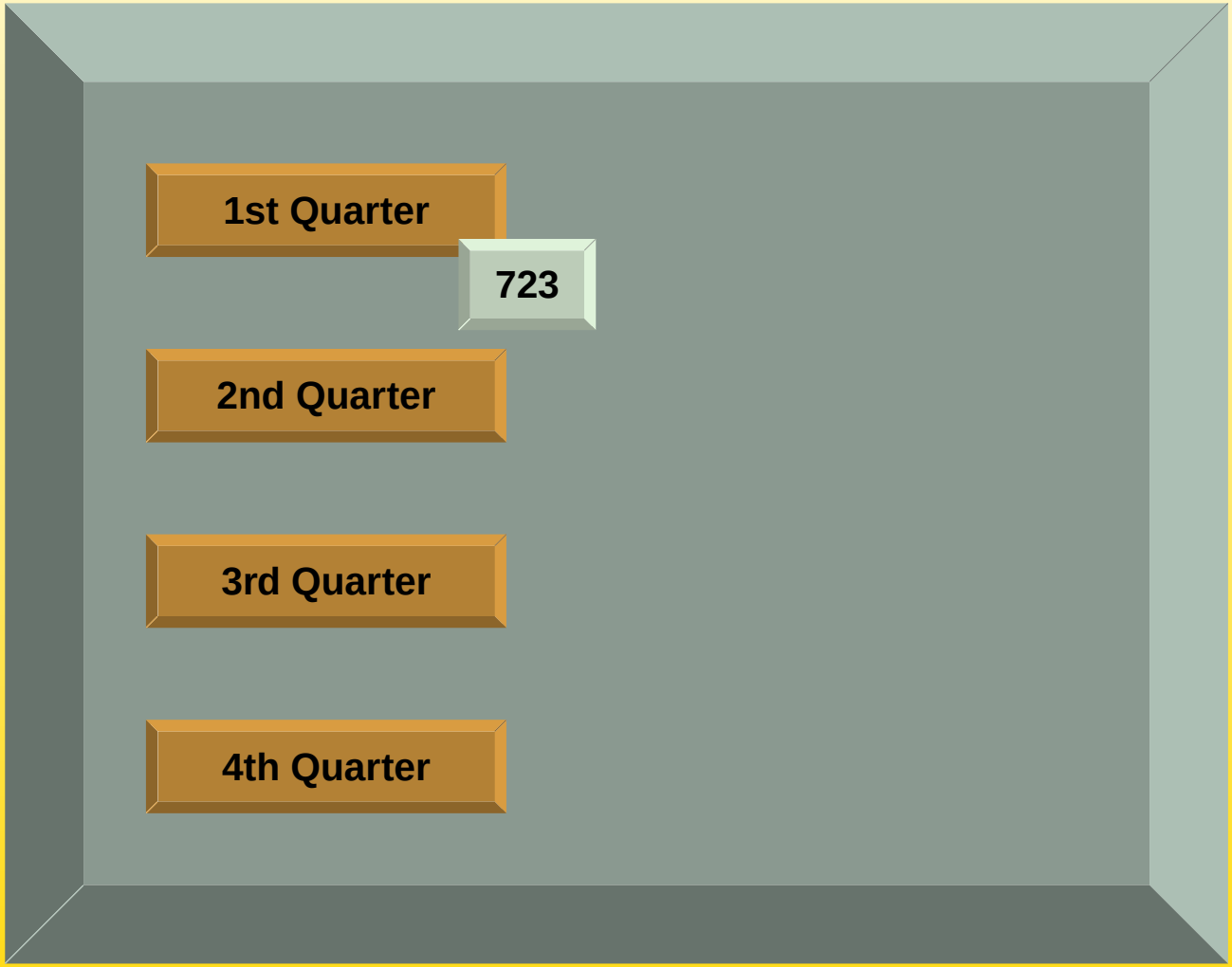
Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723
2nd Quarter	
3rd Quarter	
4th Quarter	

End of animation

4 – Results of the year

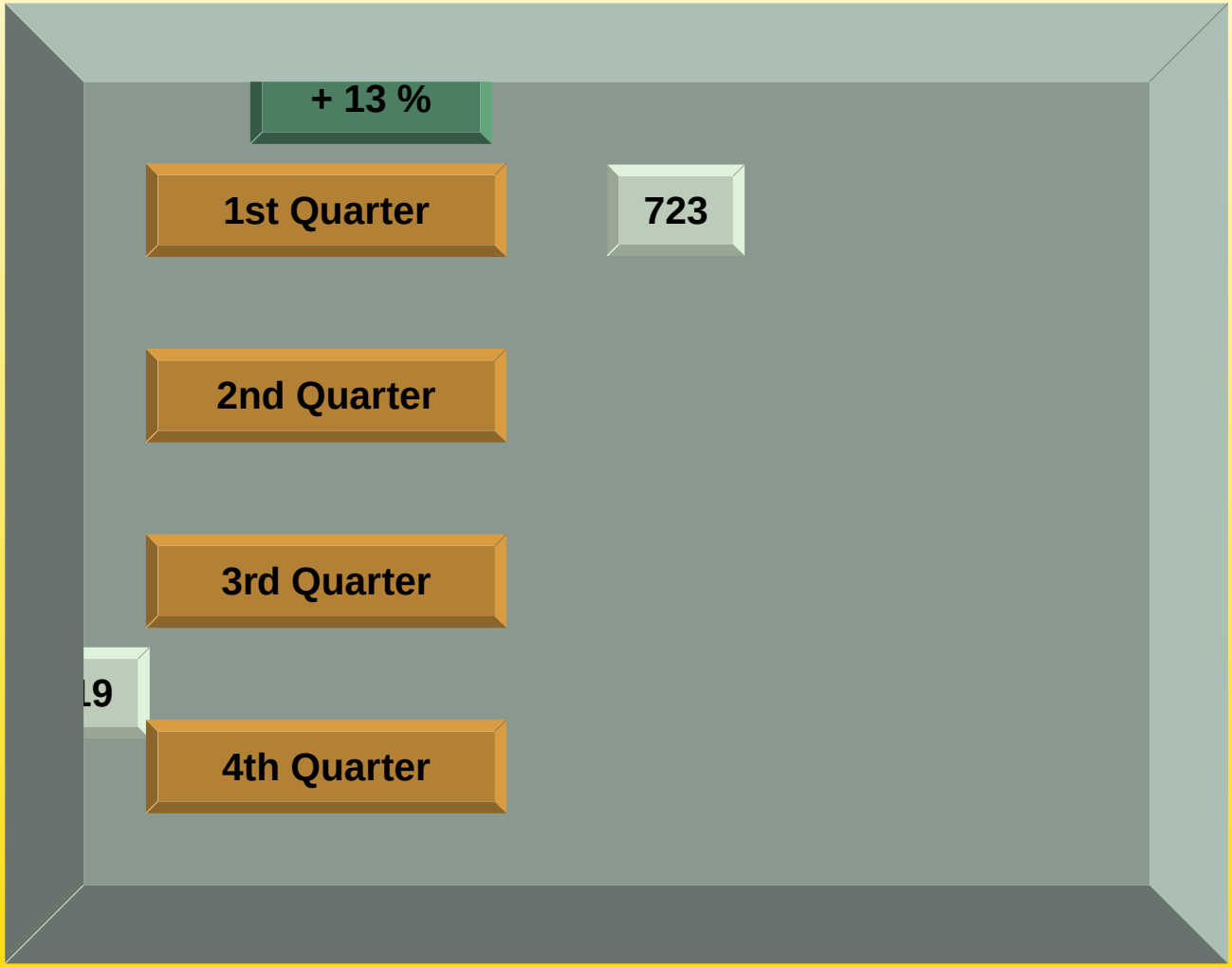
Table 1: Results of the year

1st Quarter	723
2nd Quarter	
3rd Quarter	
4th Quarter	

End of animation

4 – Results of the year

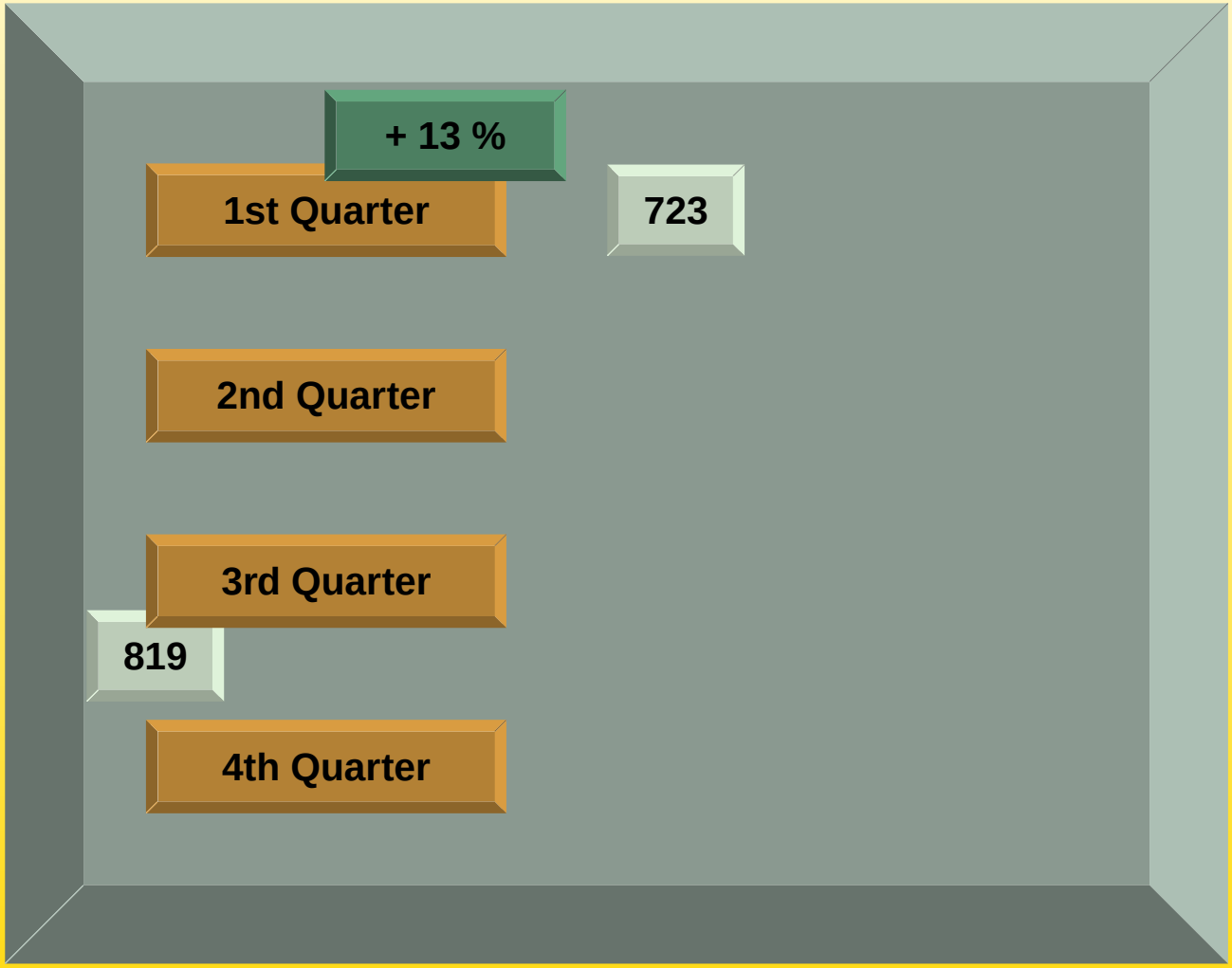
Table 1: Results of the year



End of animation

4 – Results of the year

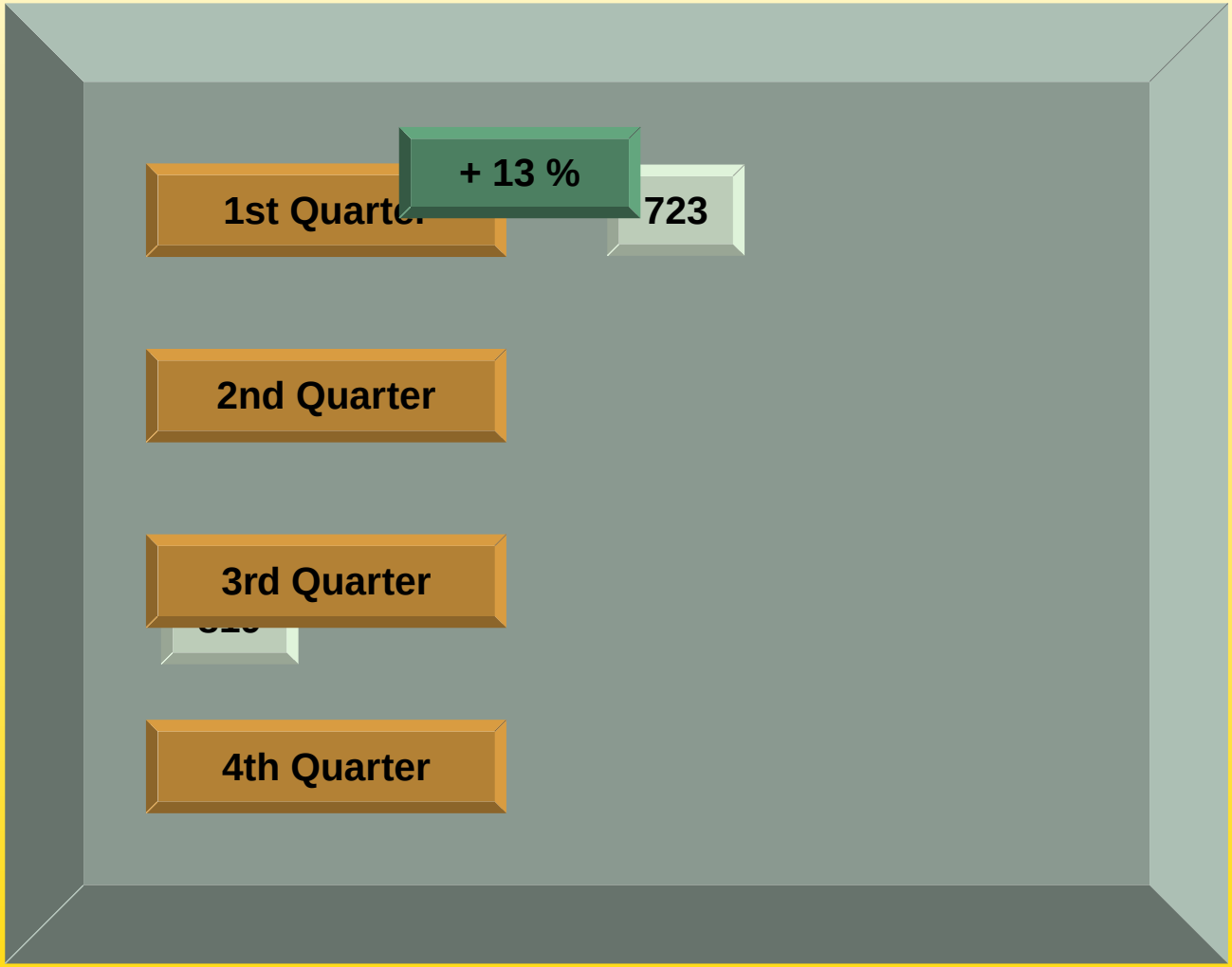
Table 1: Results of the year



End of animation

4 – Results of the year

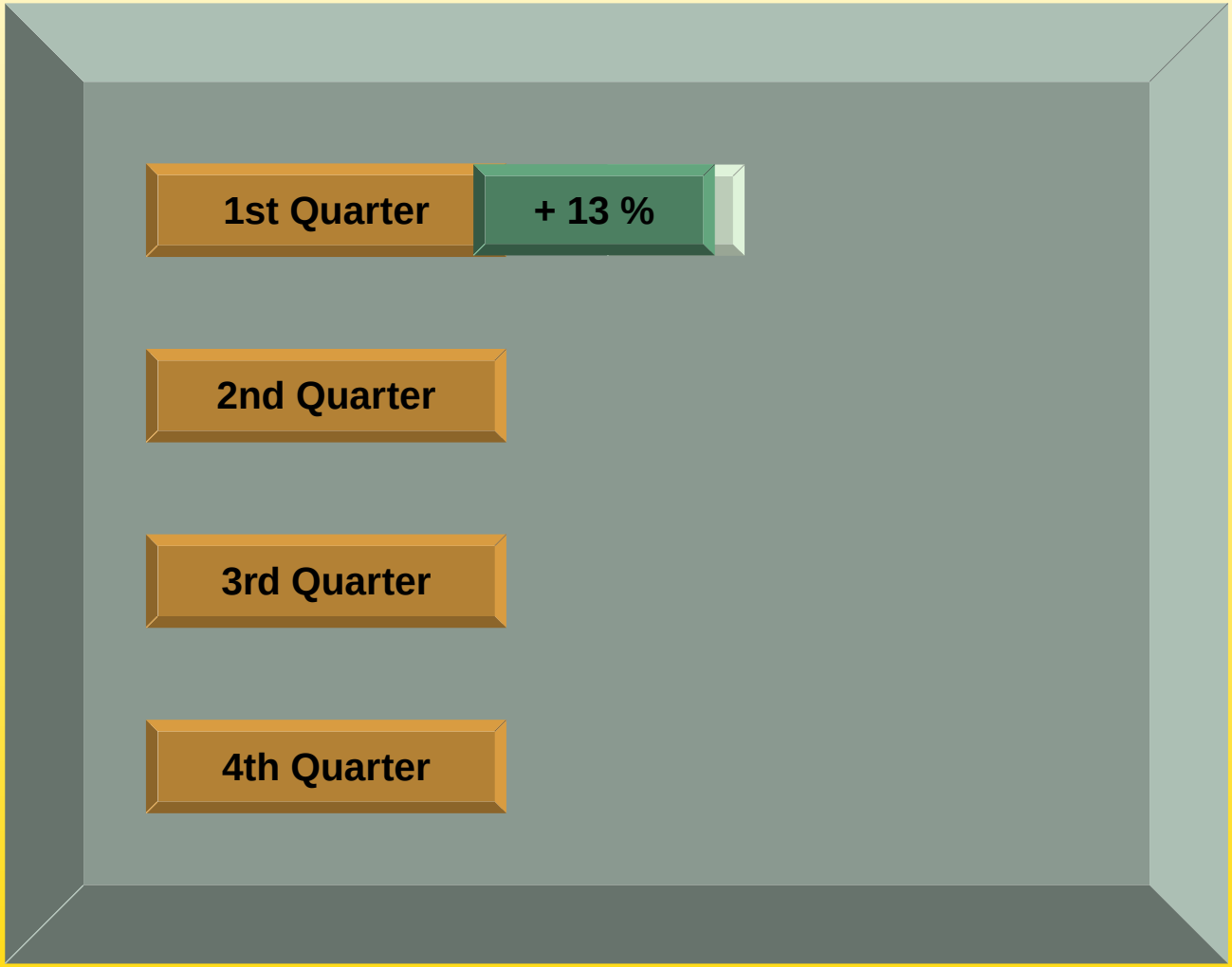
Table 1: Results of the year



End of animation

4 – Results of the year

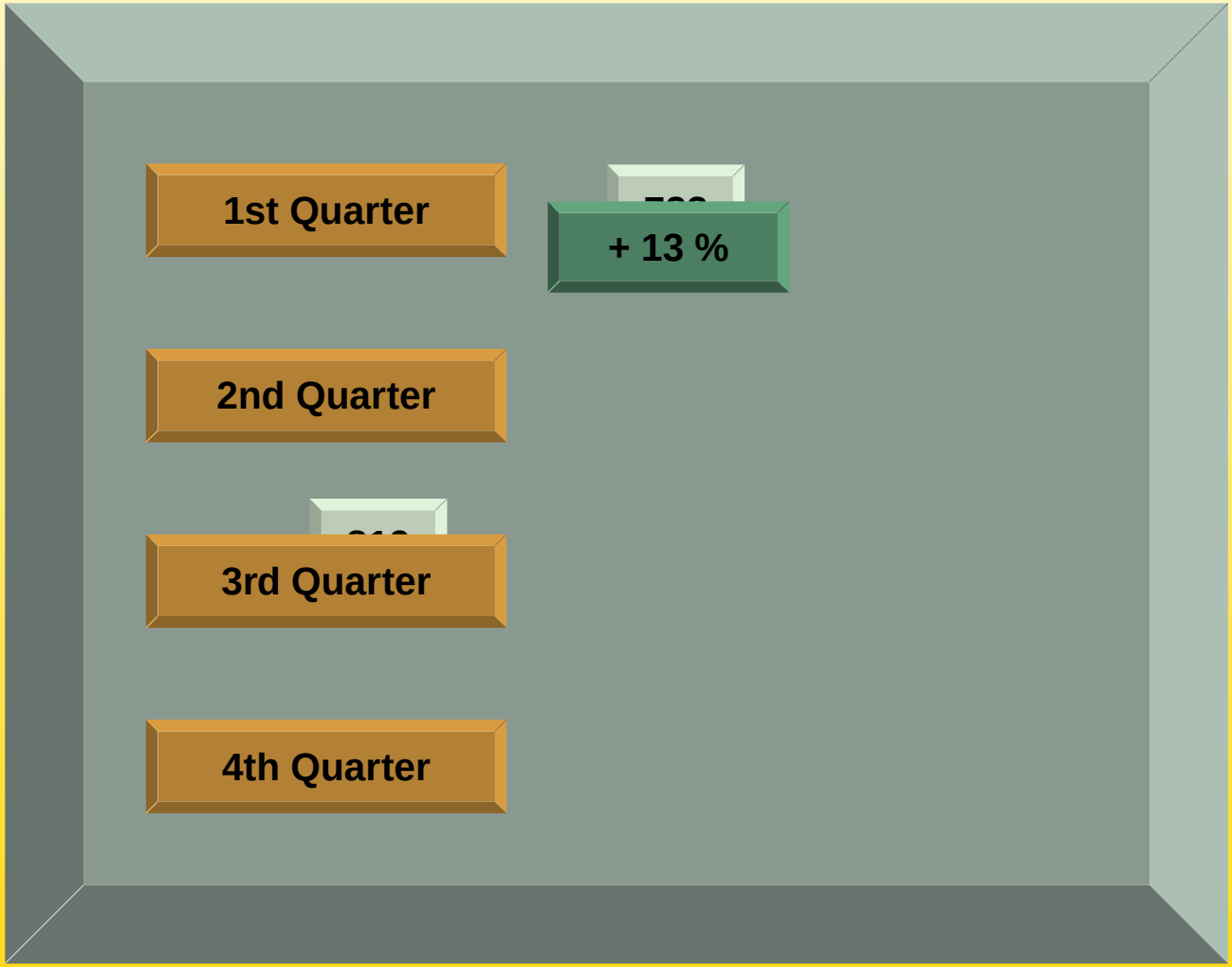
Table 1: Results of the year



End of animation

4 – Results of the year

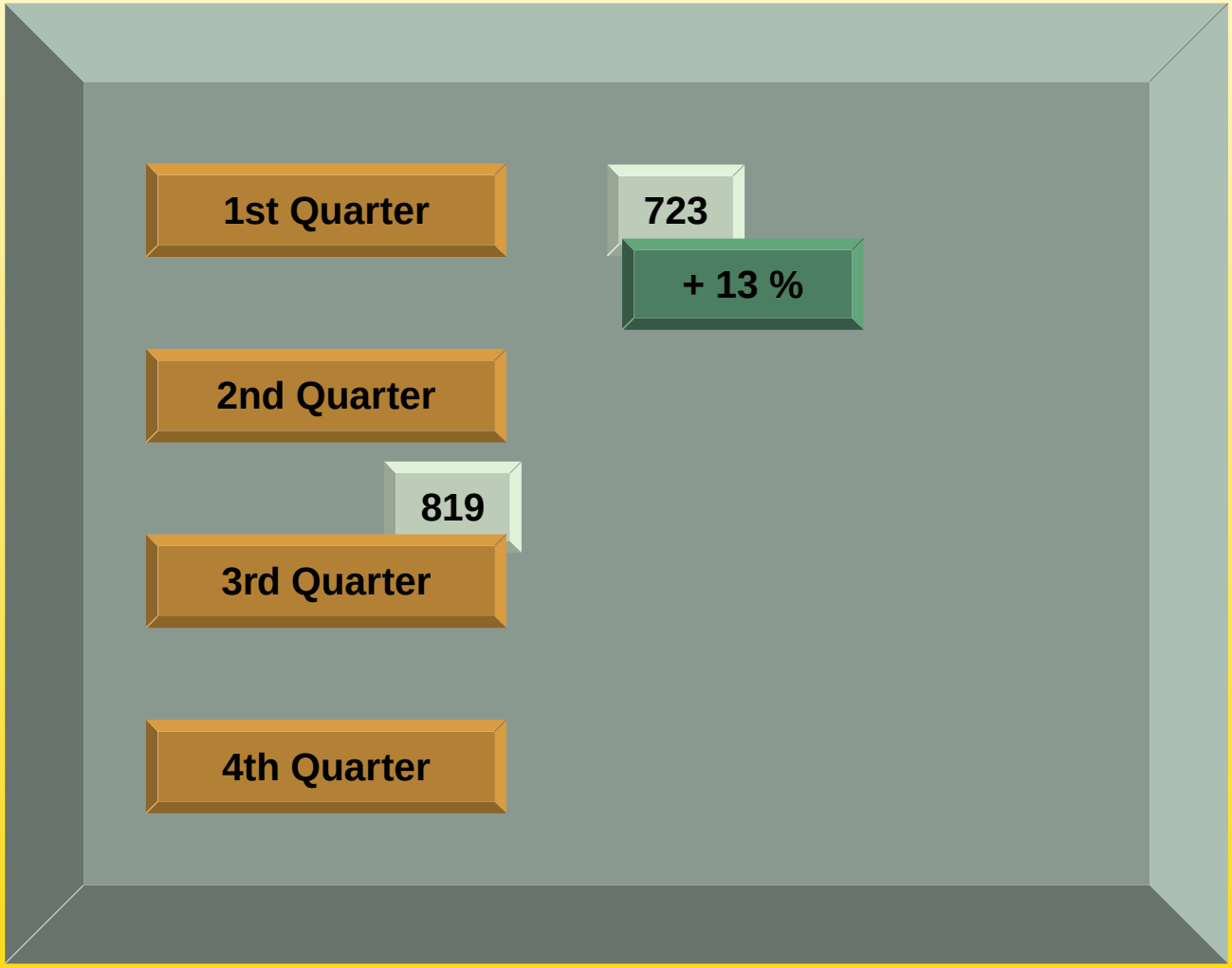
Table 1: Results of the year



End of animation

4 – Results of the year

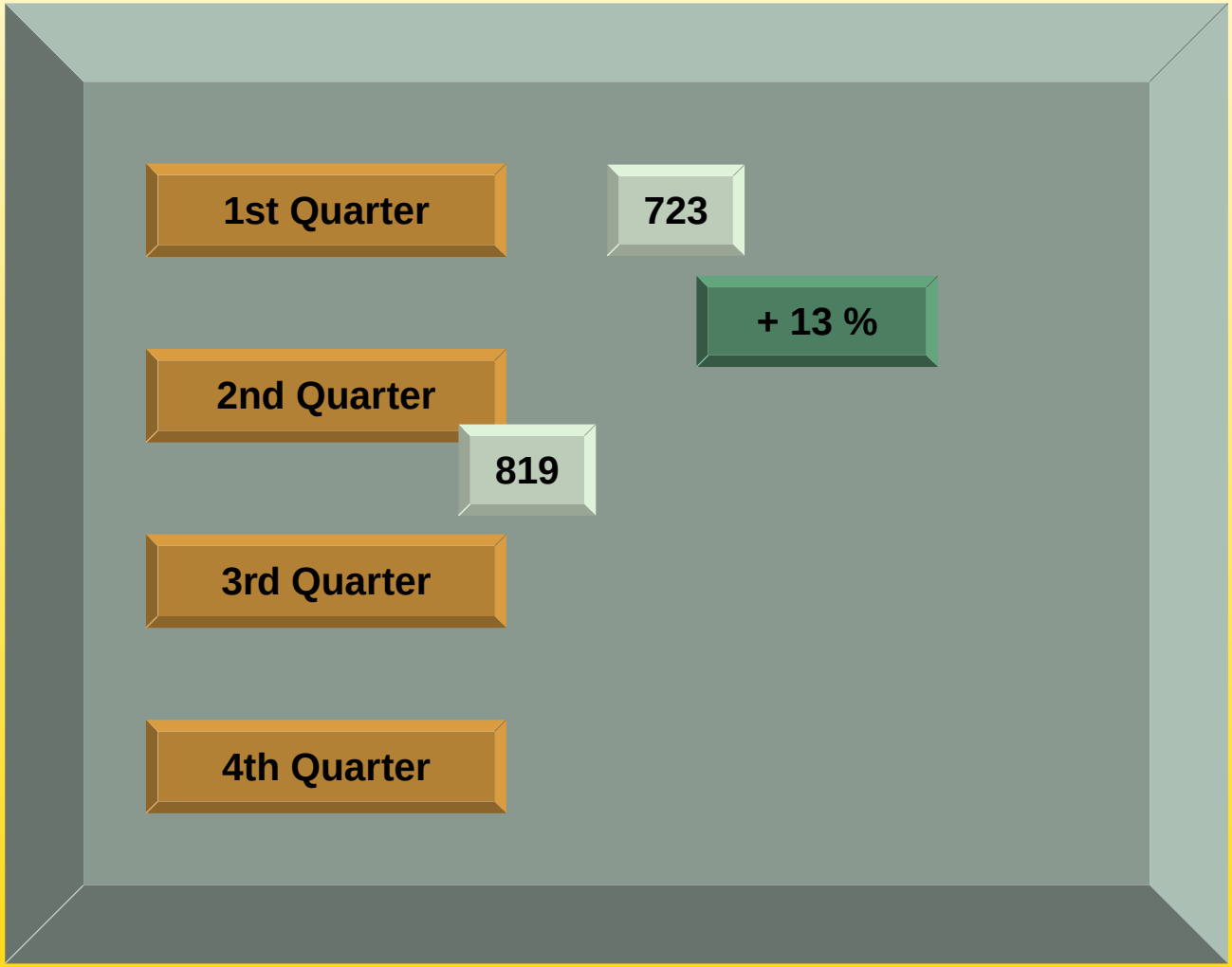
Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter		
4th Quarter		

End of animation

4 – Results of the year

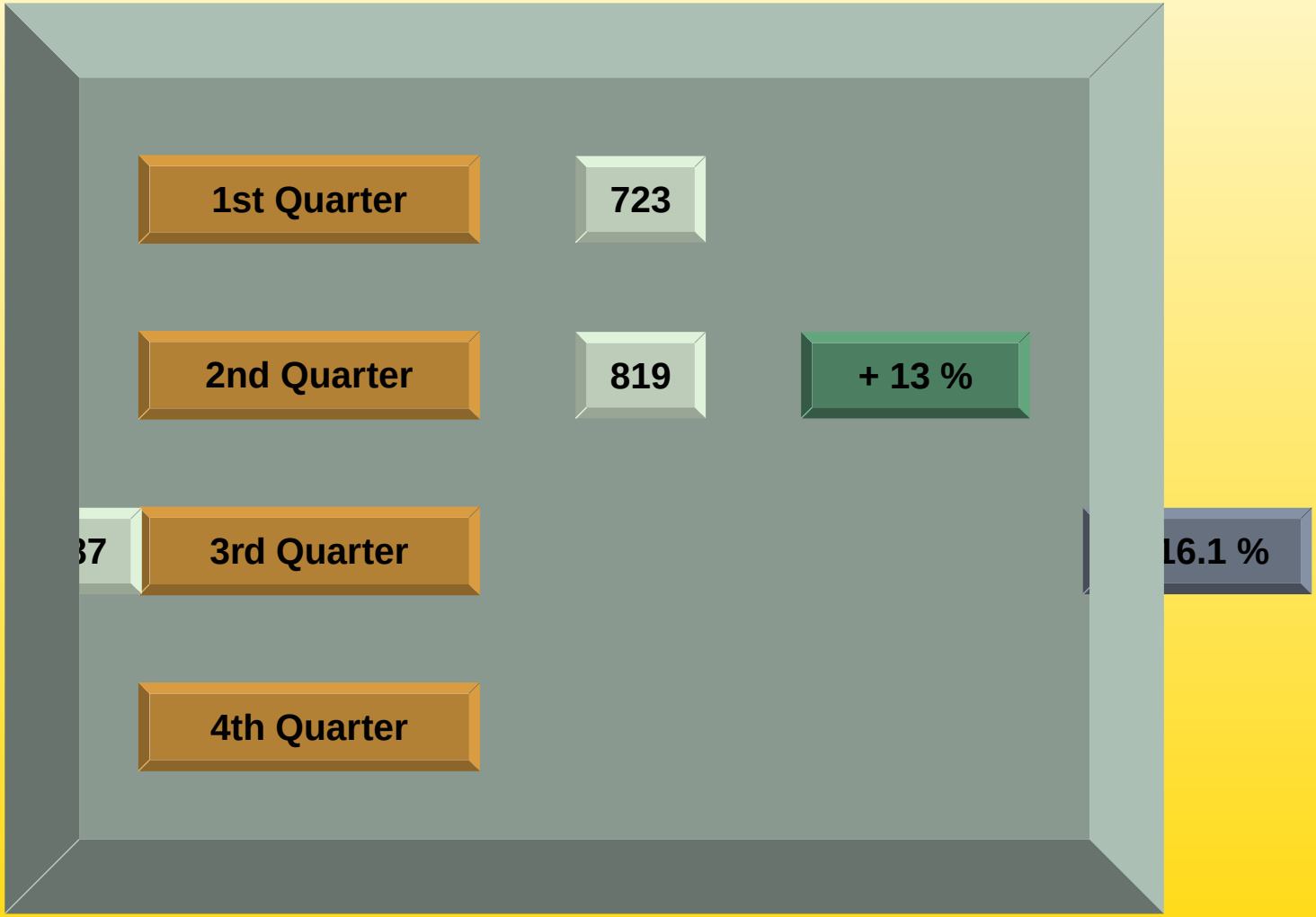
Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter		
4th Quarter		

End of animation

4 – Results of the year

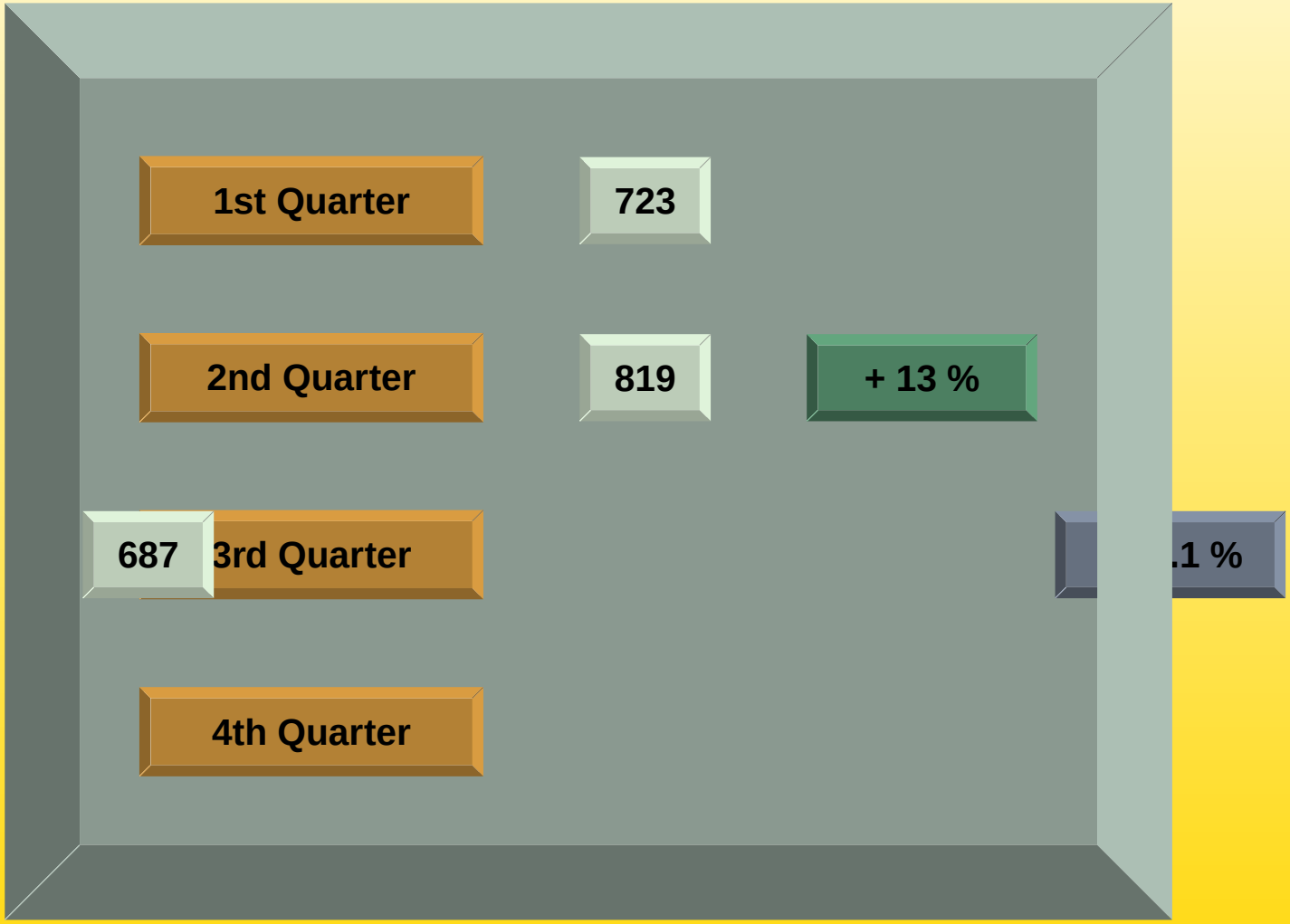
Table 1: Results of the year



End of animation

4 – Results of the year

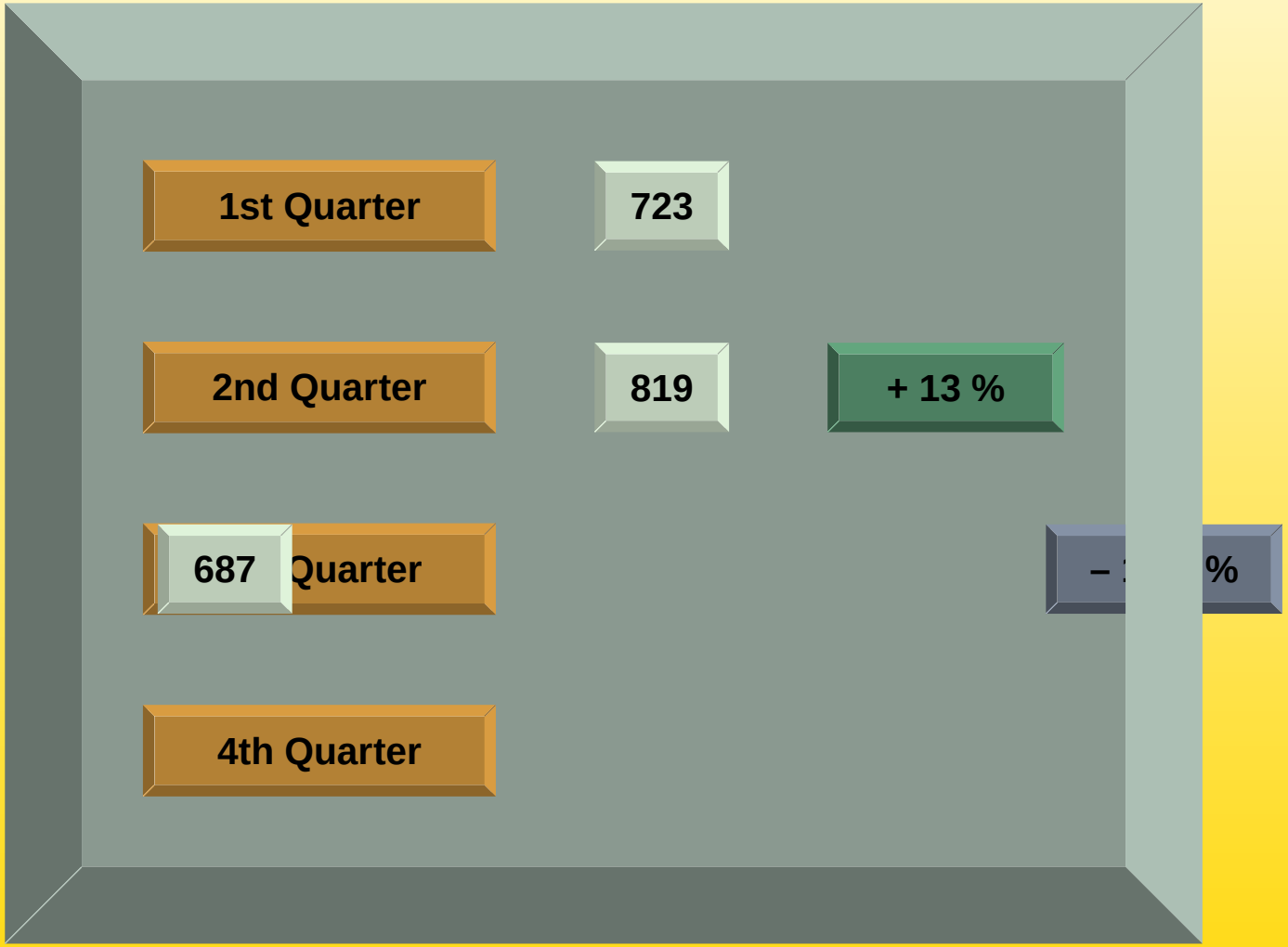
Table 1: Results of the year



End of animation

4 – Results of the year

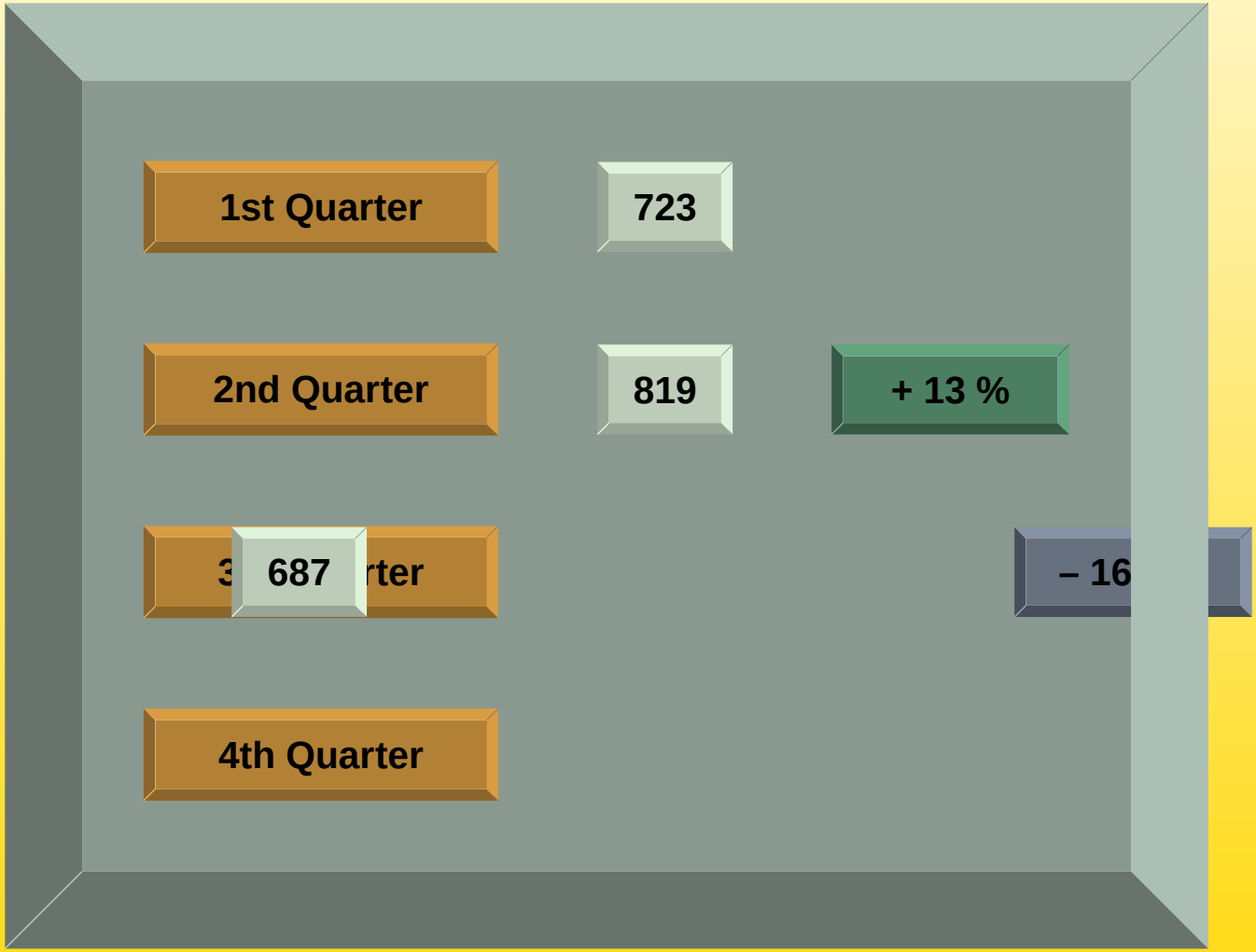
Table 1: Results of the year



End of animation

4 – Results of the year

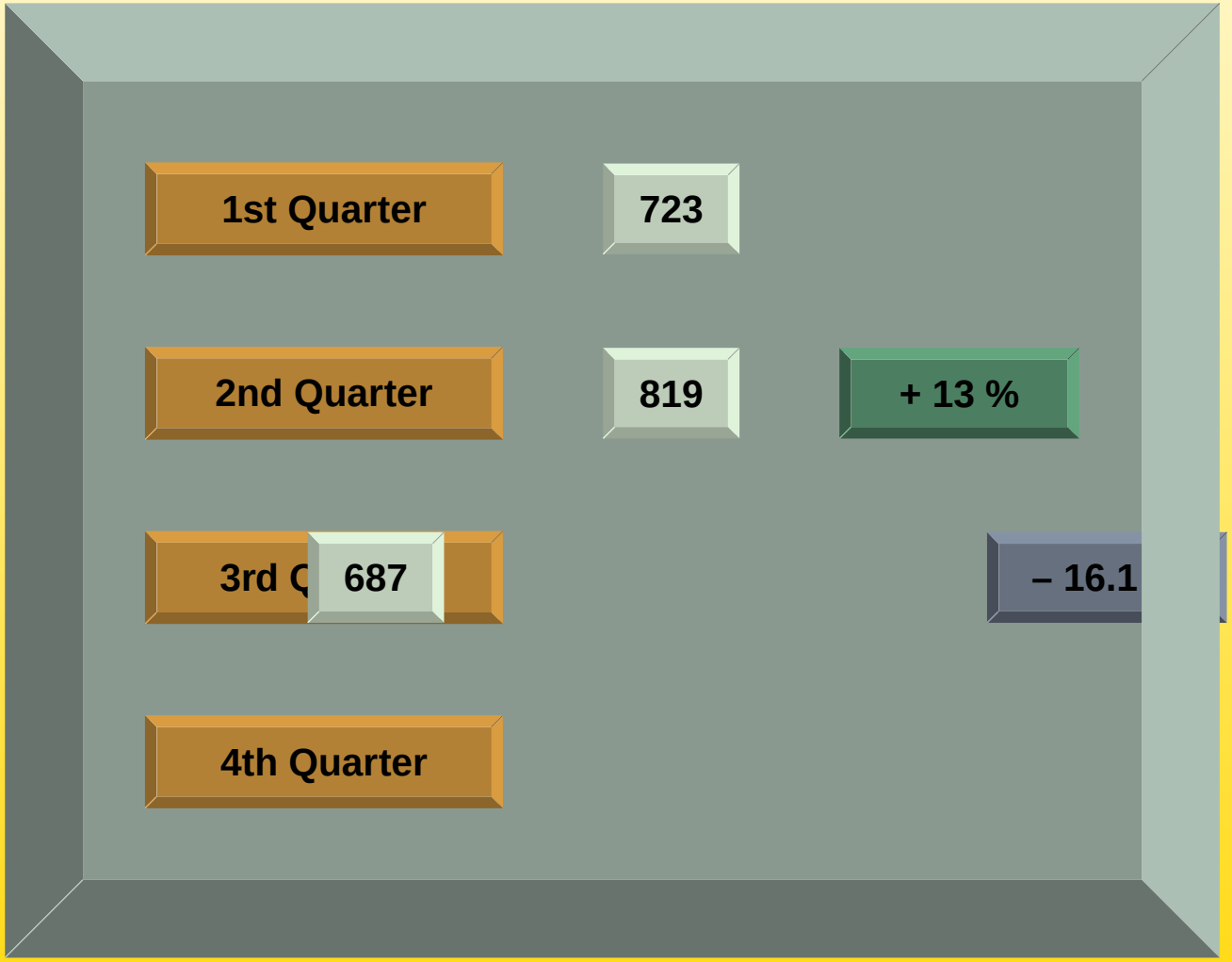
Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year



End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quar	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	894	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	894	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	894 819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	894	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
	894	
3rd Quarter	687	– 16.1 %
4th Quarter		
		+ 30.1 %

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	894	– 16.1 %
4th Quarter		
		+ 30.1 %

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	894	– 16.1 %
4th Quarter		

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter	894	+ 30.1 %

End of animation

4 – Results of the year

Table 1: Results of the year

1st Quarter	723	
2nd Quarter	819	+ 13 %
3rd Quarter	687	– 16.1 %
4th Quarter	894	+ 30.1 %

End of animation

5 – Clock

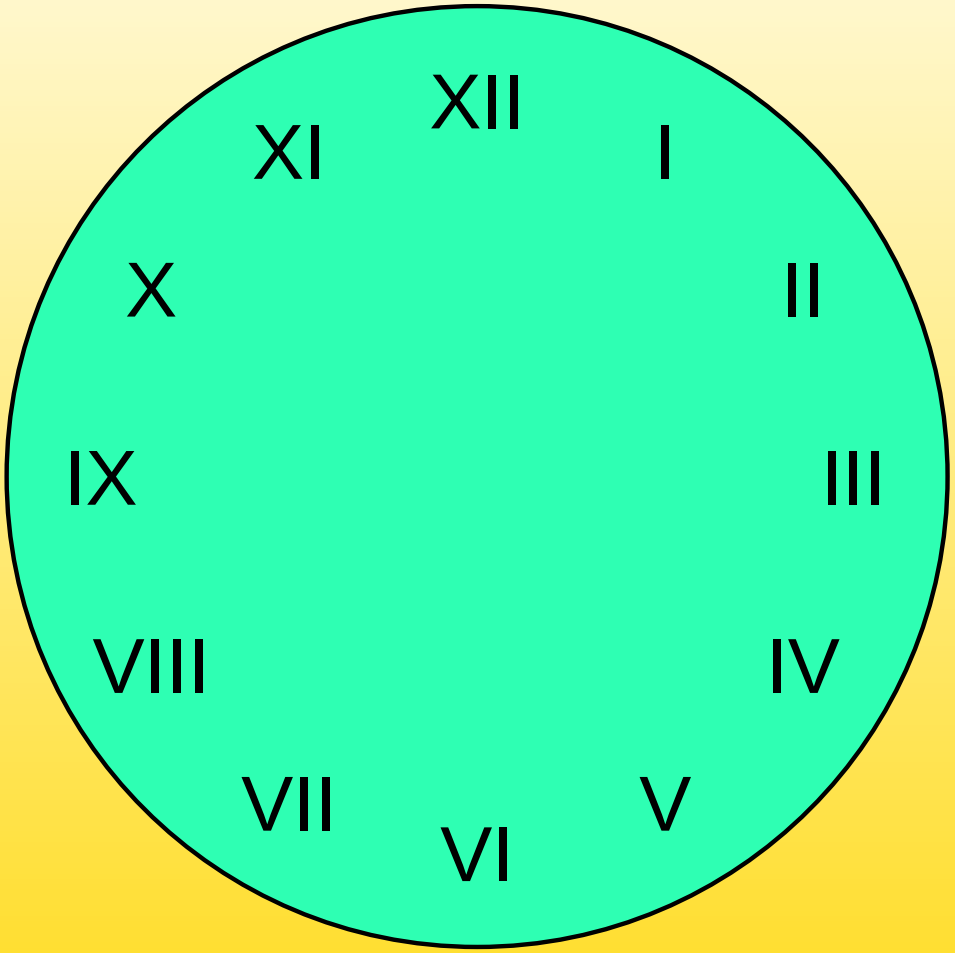


Figure 3: Clock

End of animation

5 – Clock

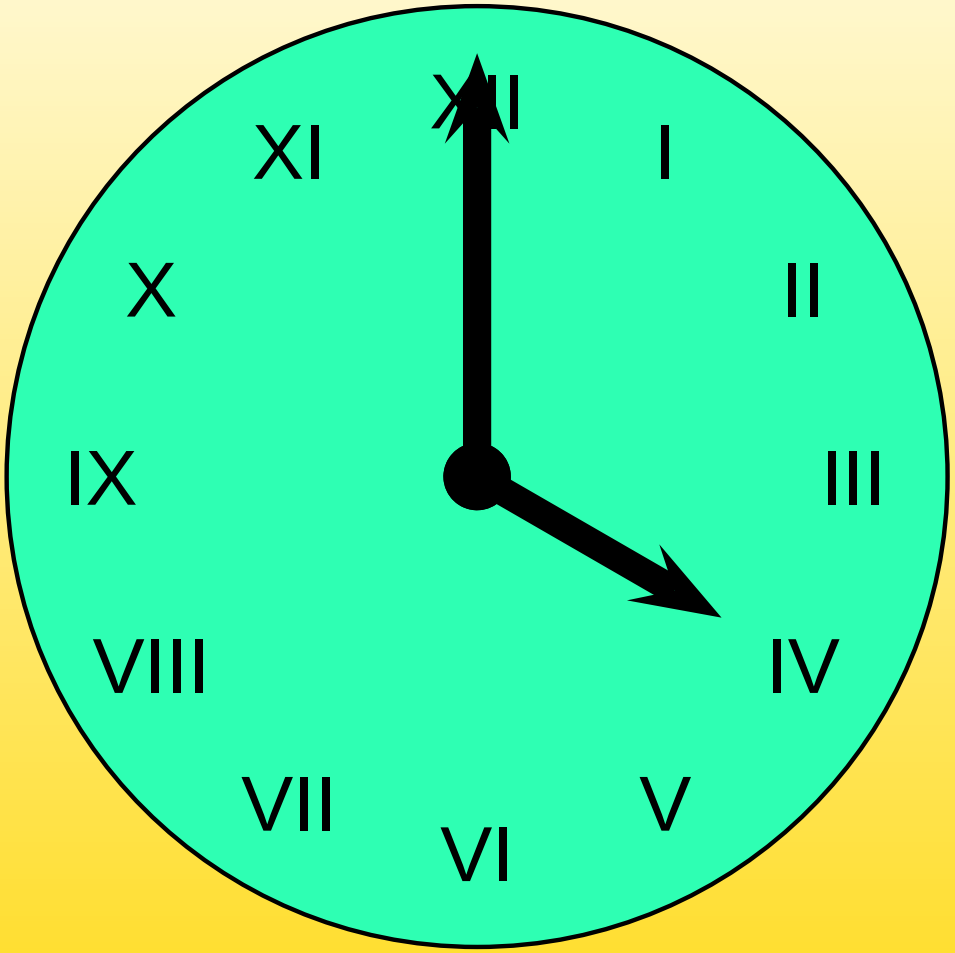


Figure 3: Clock

End of animation

5 – Clock

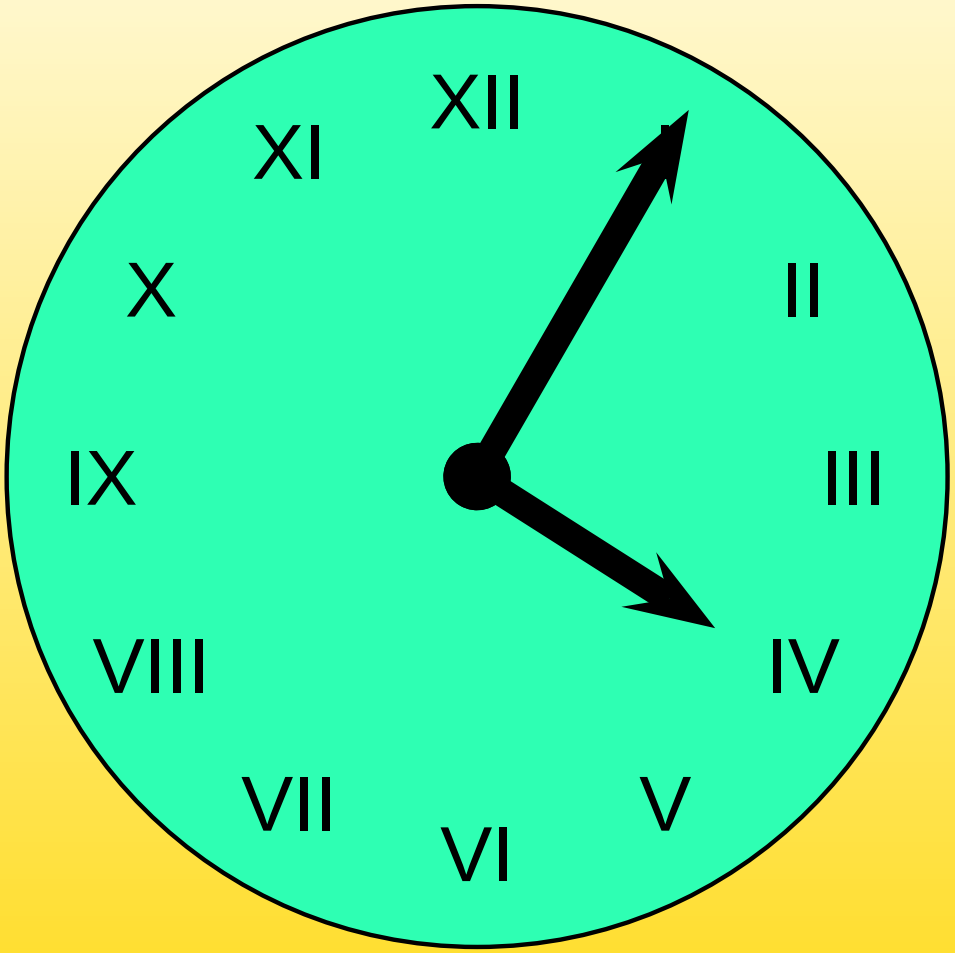


Figure 3: Clock

End of animation

5 – Clock

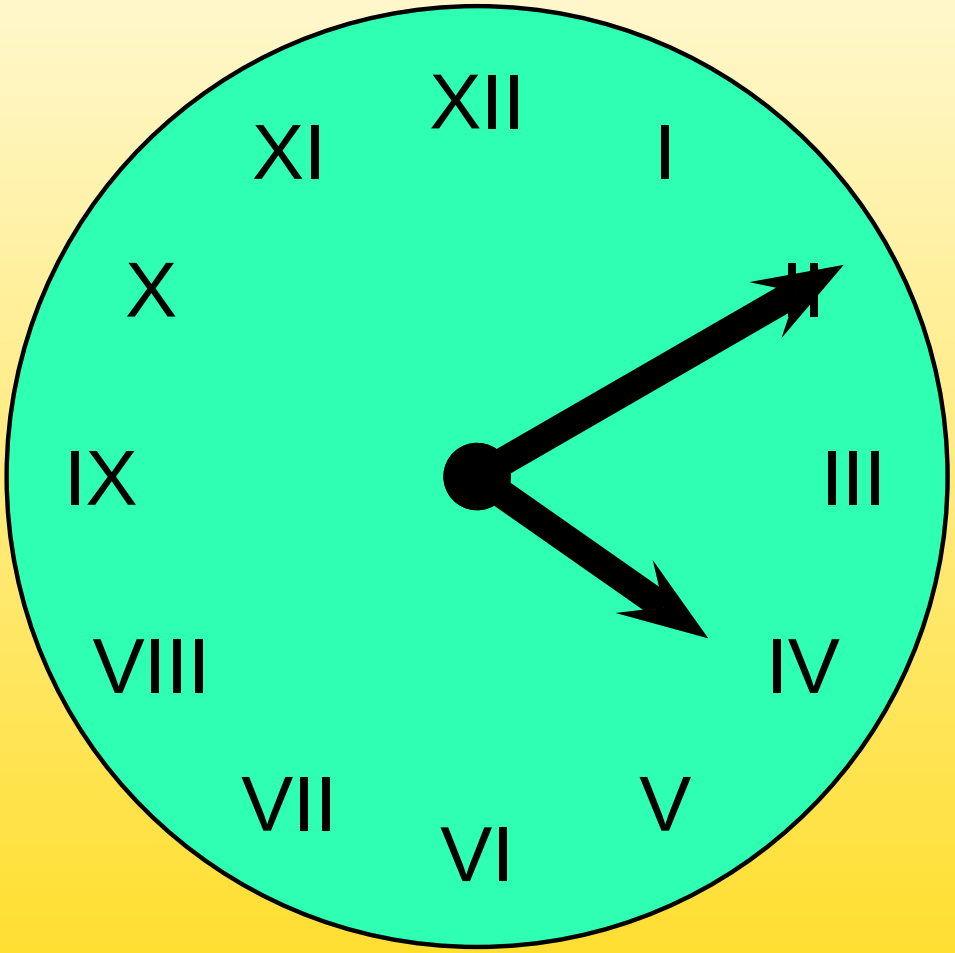


Figure 3: Clock

End of animation

5 – Clock

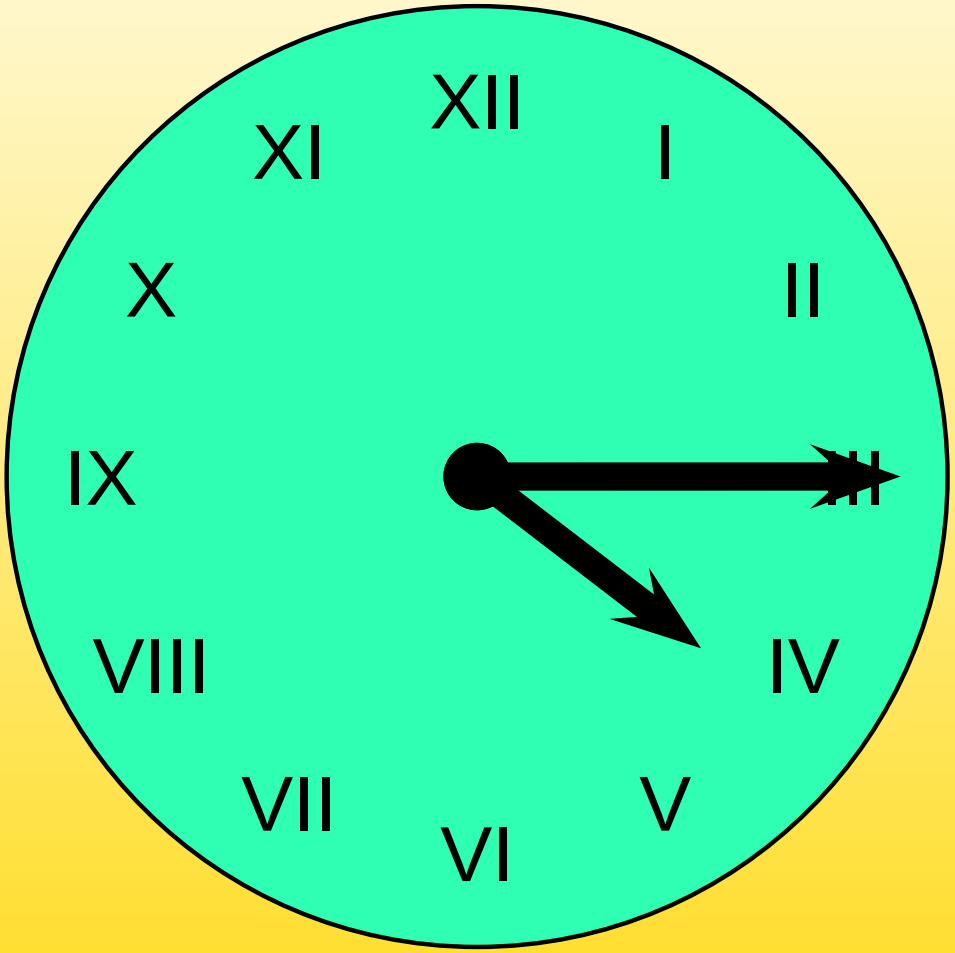


Figure 3: Clock

End of animation

5 – Clock

Figure 3: Clock

End of animation

5 – Clock

Figure 3: Clock

End of animation

5 – Clock

Figure 3: Clock

End of animation

5 – Clock

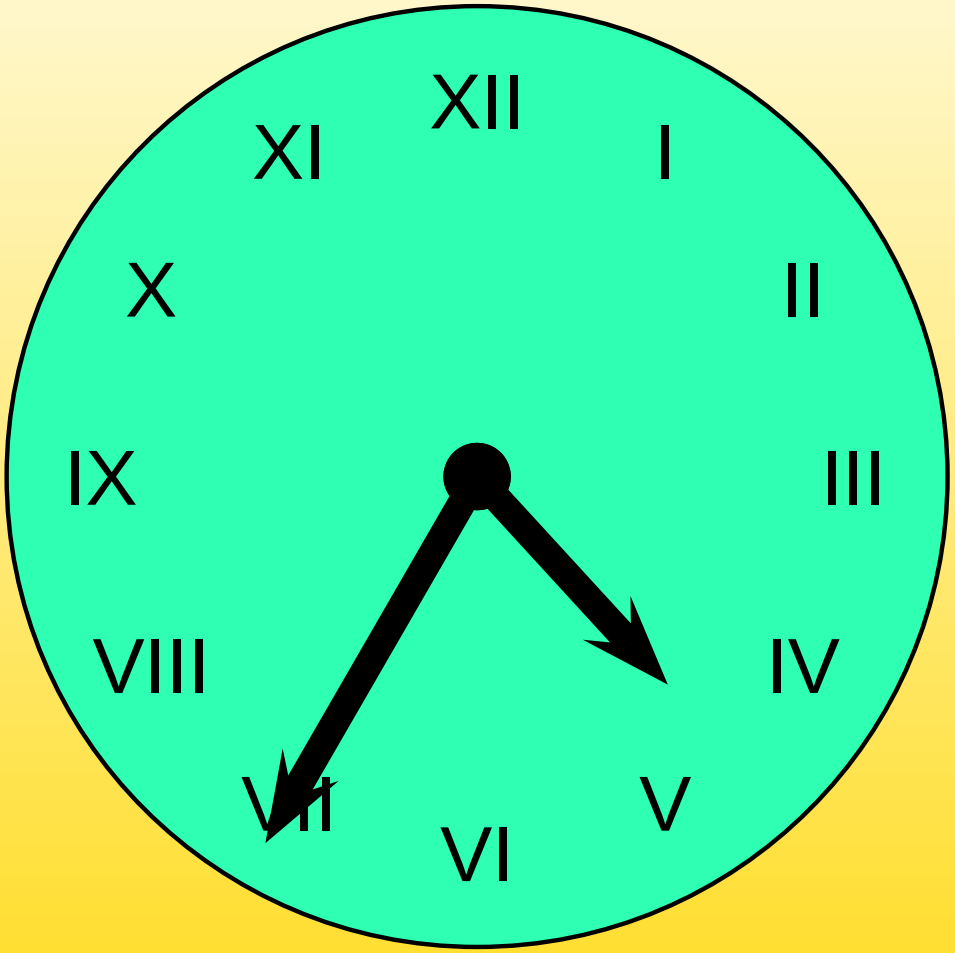


Figure 3: Clock

End of animation

5 – Clock

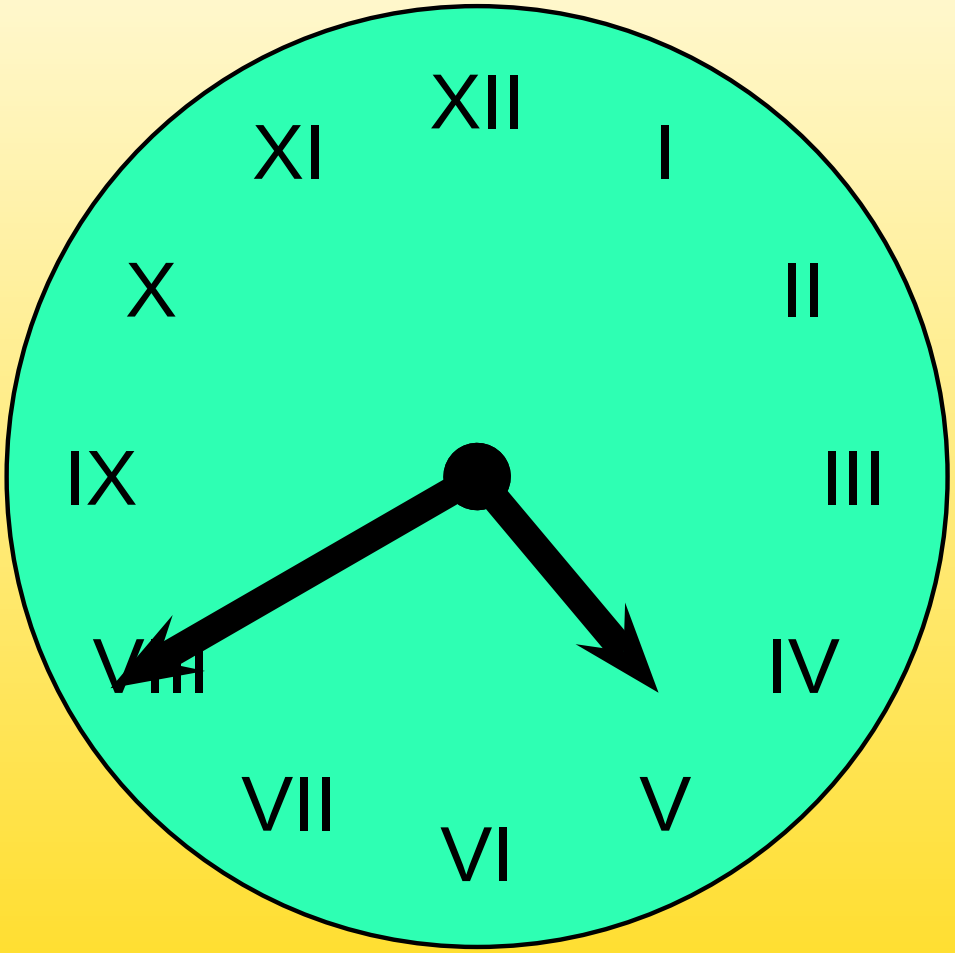


Figure 3: Clock

End of animation

5 – Clock

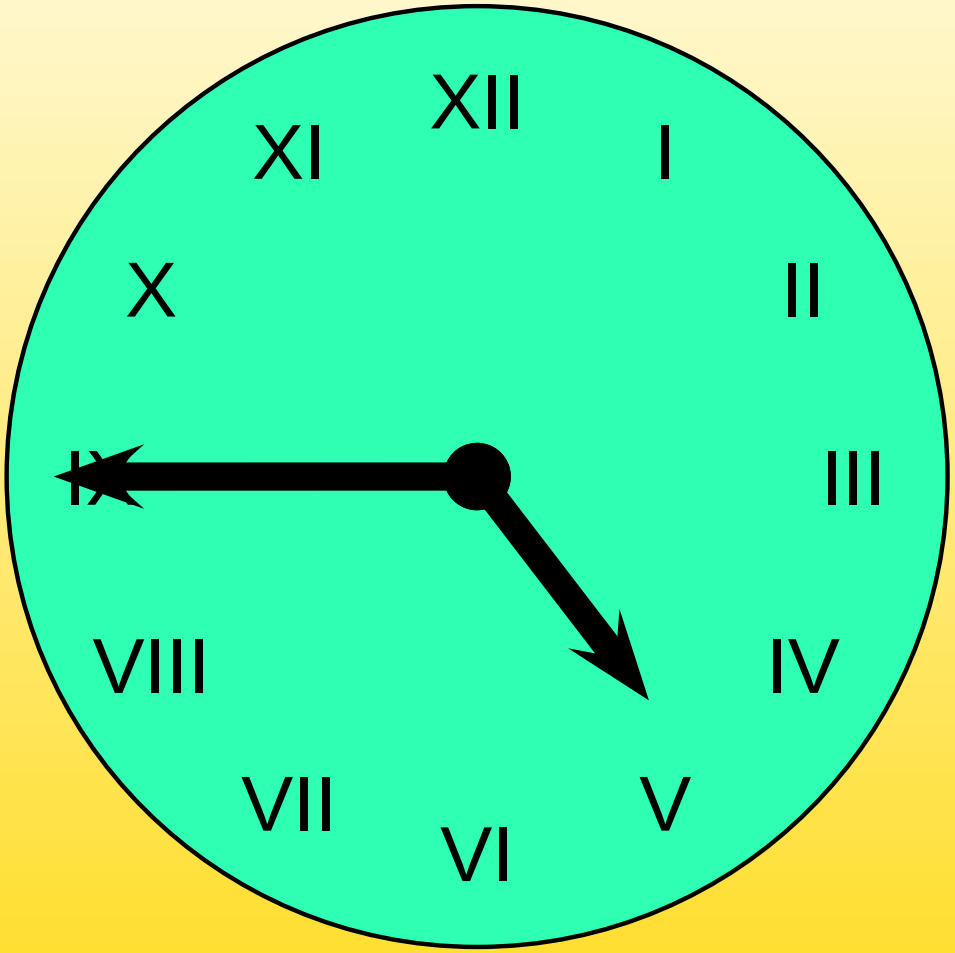


Figure 3: Clock

End of animation

5 – Clock

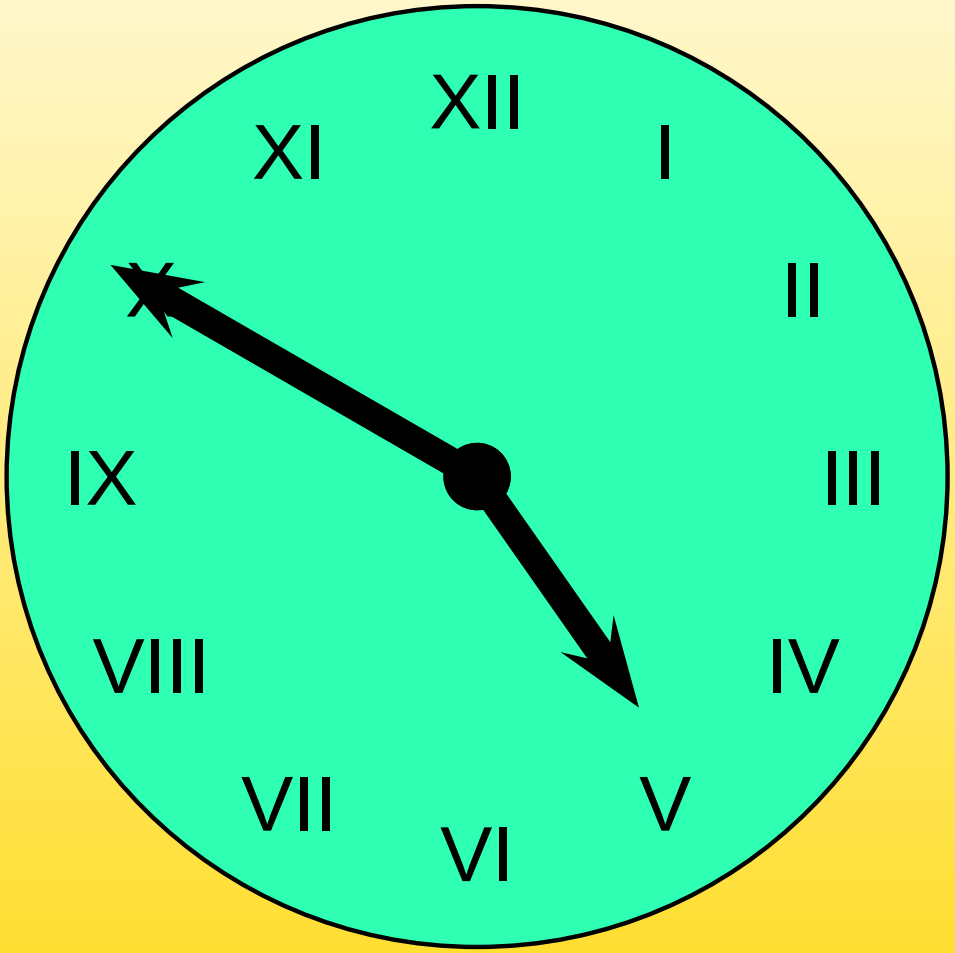


Figure 3: Clock

End of animation

5 – Clock

Figure 3: Clock

End of animation

5 – Clock

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

Demonstration of animated graphics

Figure 3: Clock

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m

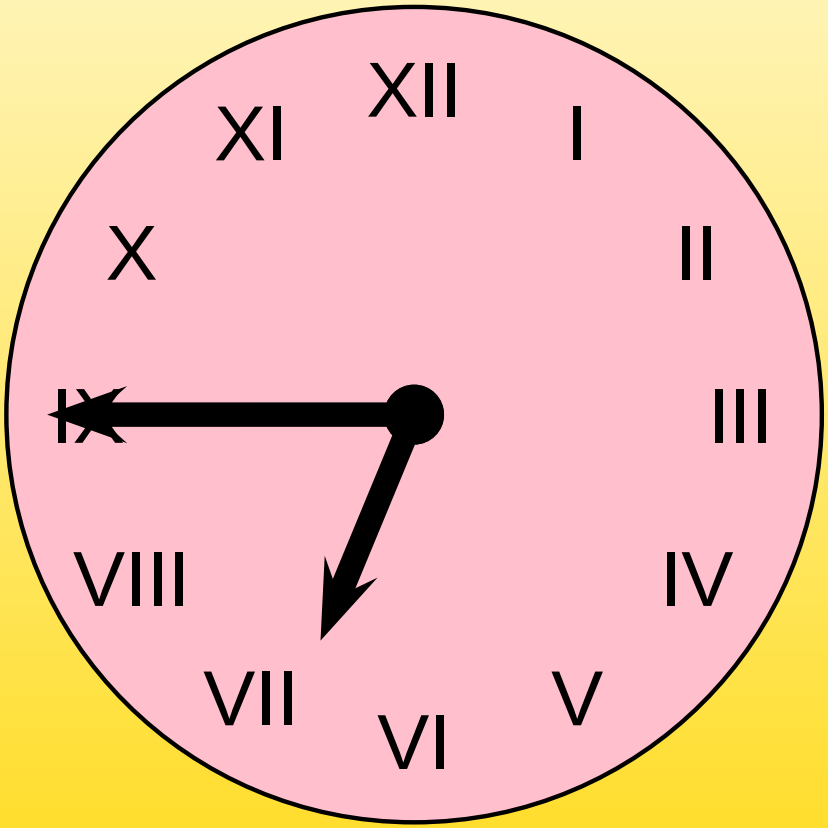


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 01s

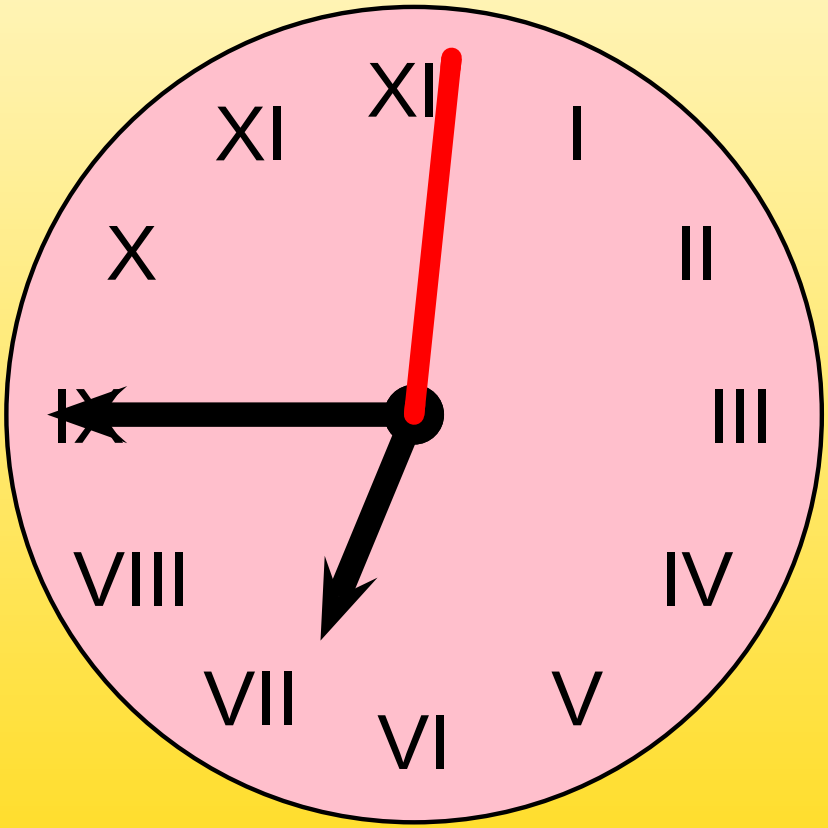


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 02s

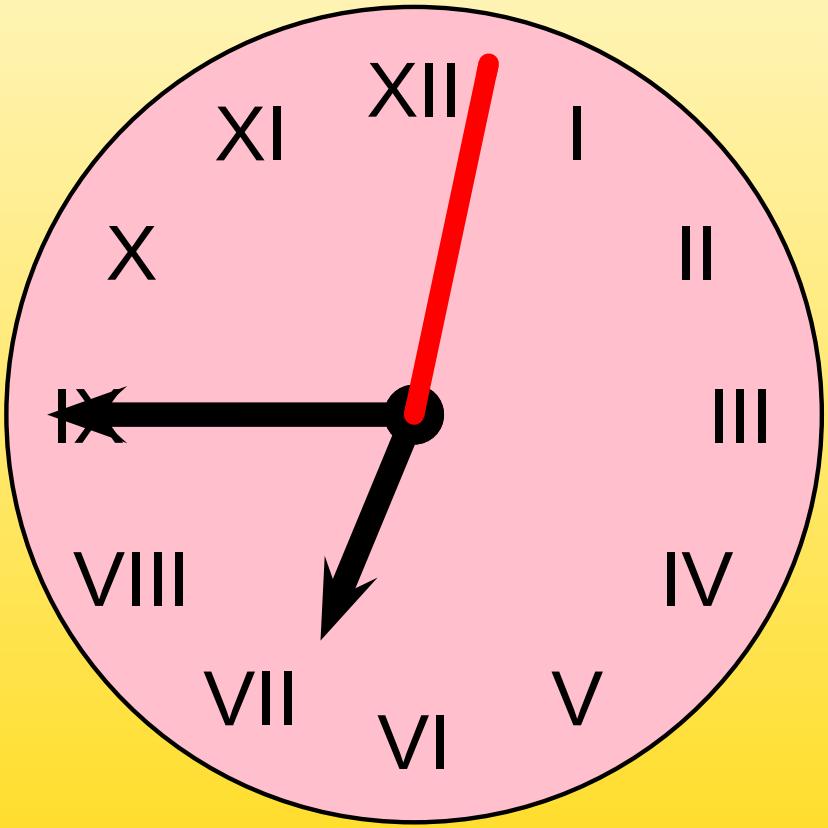


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 03s

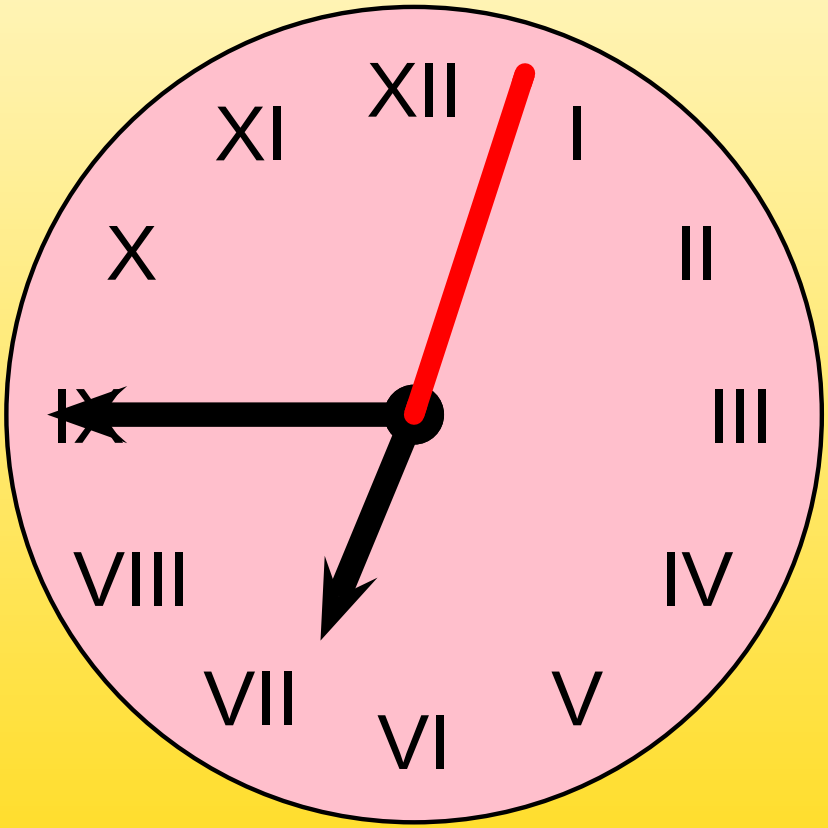


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 04s

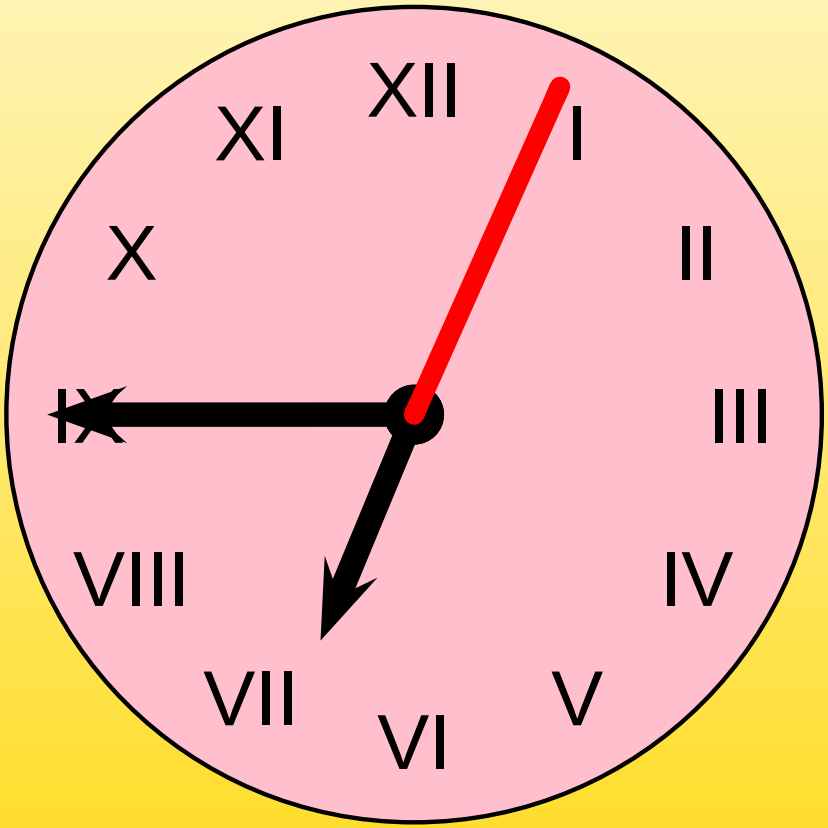


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 05s

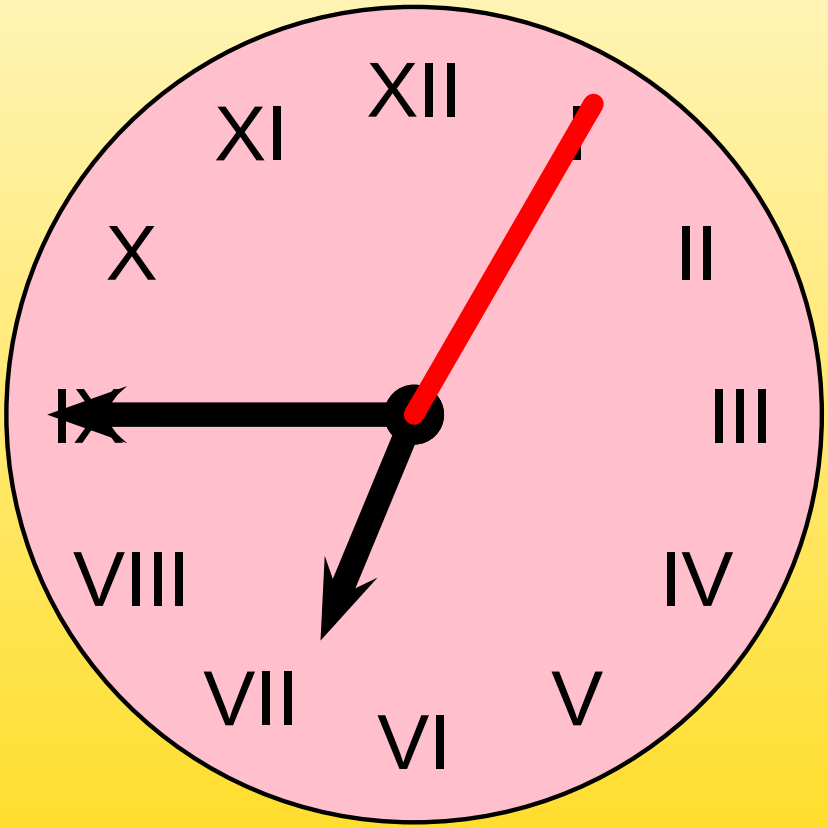


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 06s

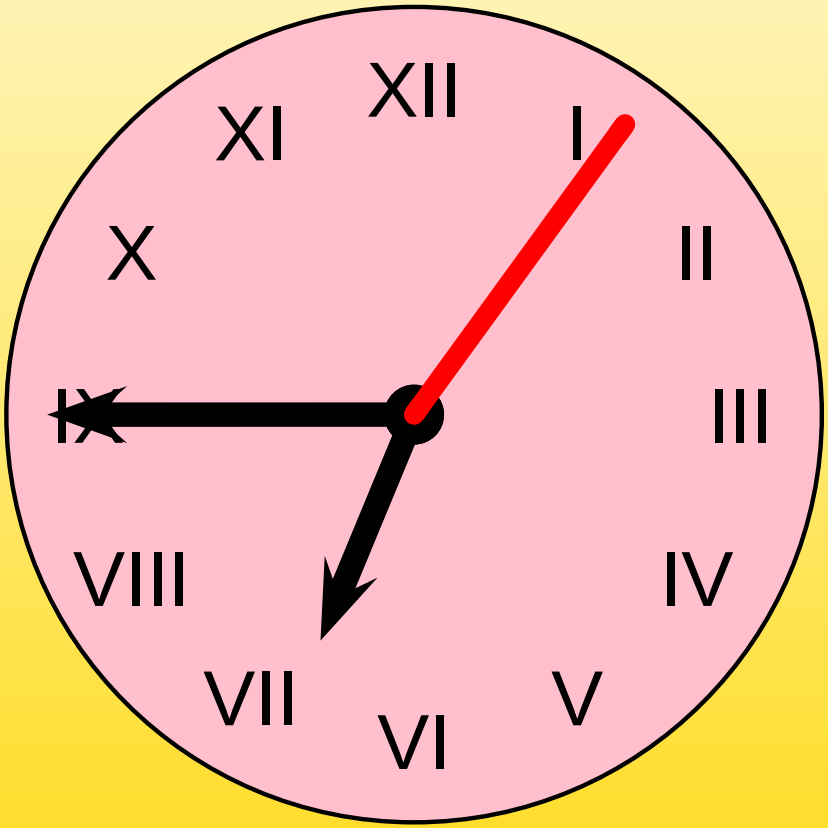


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 07s

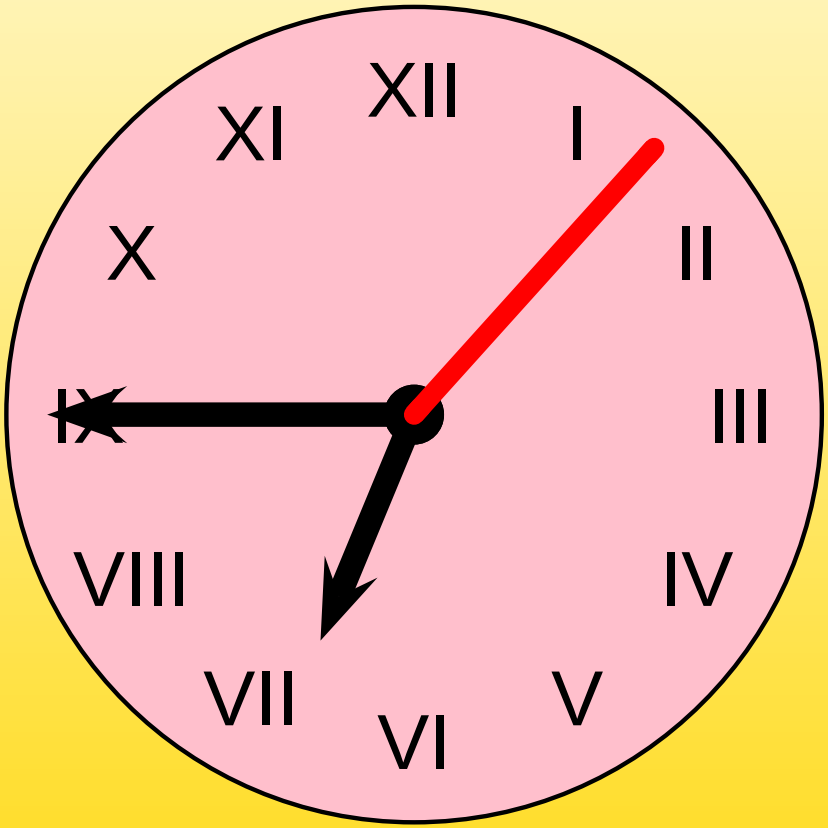


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 08s

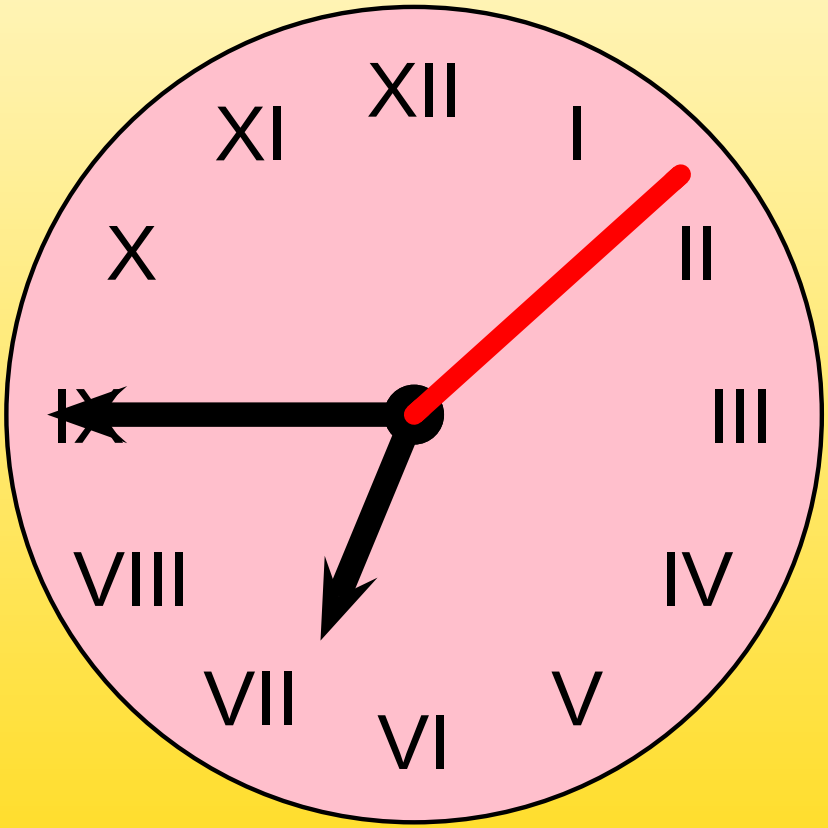


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 09s

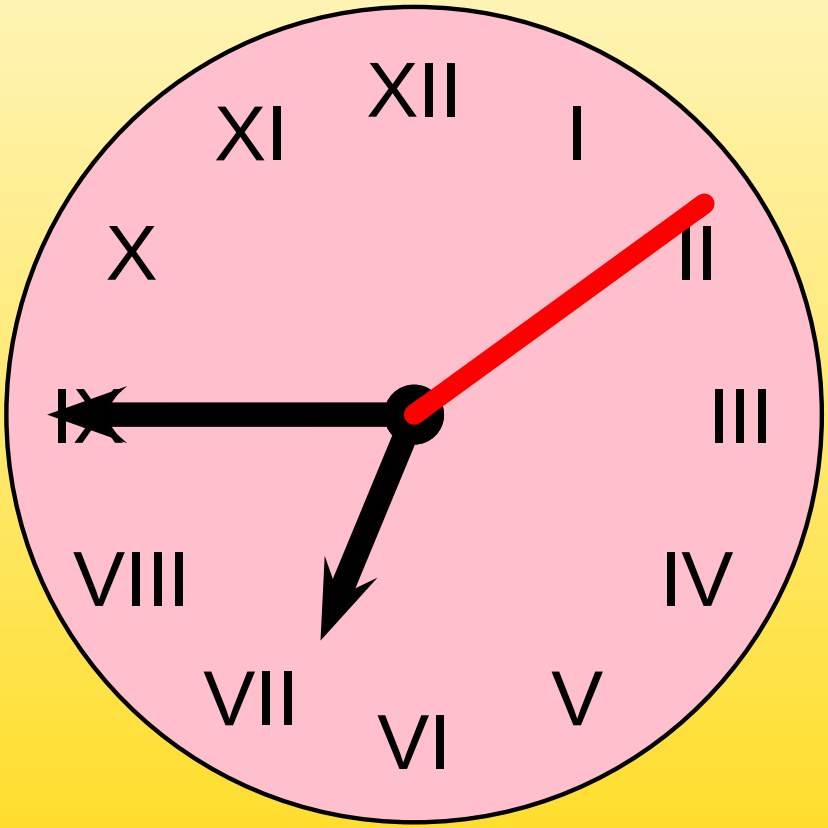


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 10s

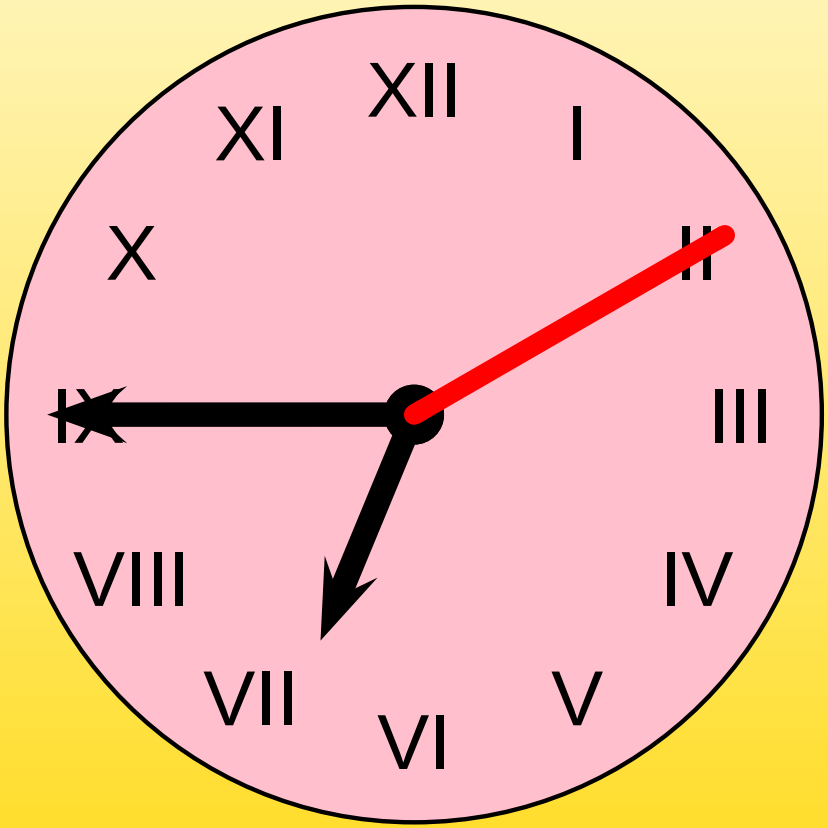


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 11s

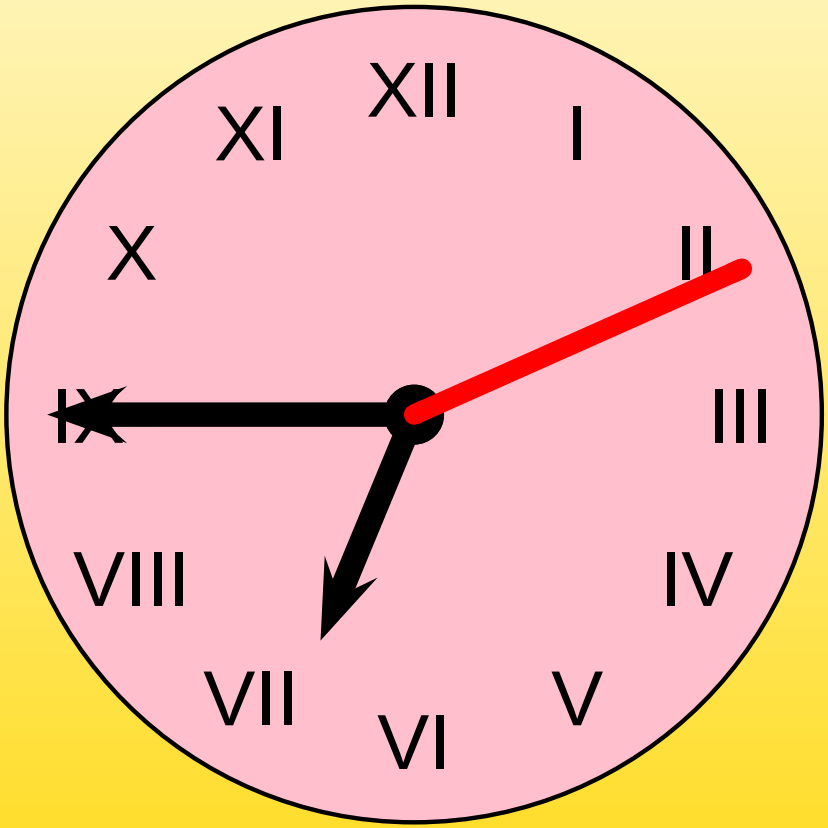


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 12s

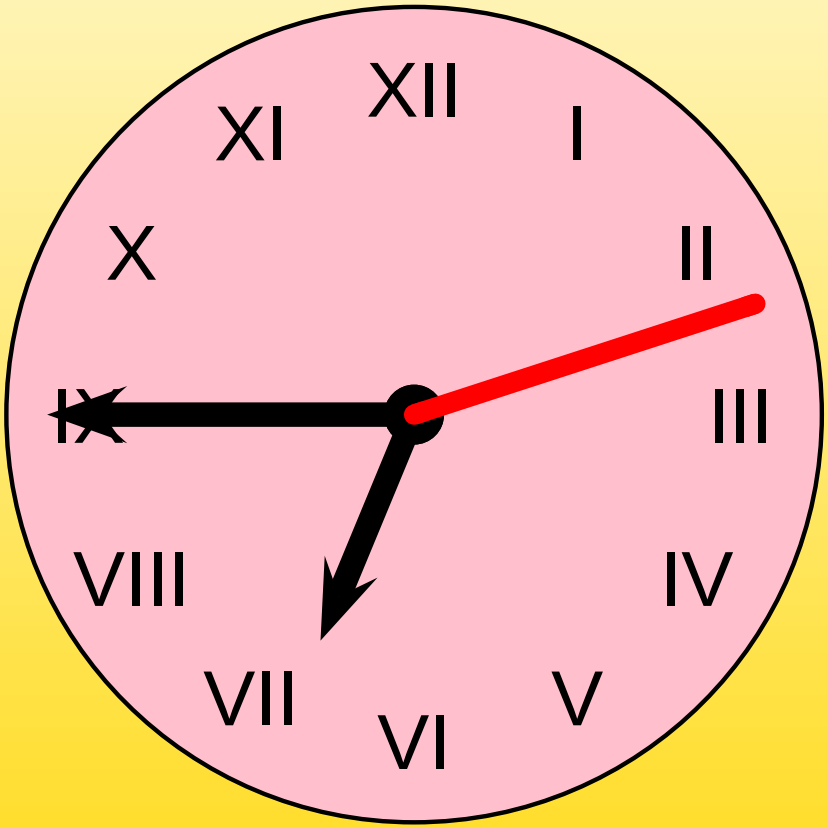


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 13s

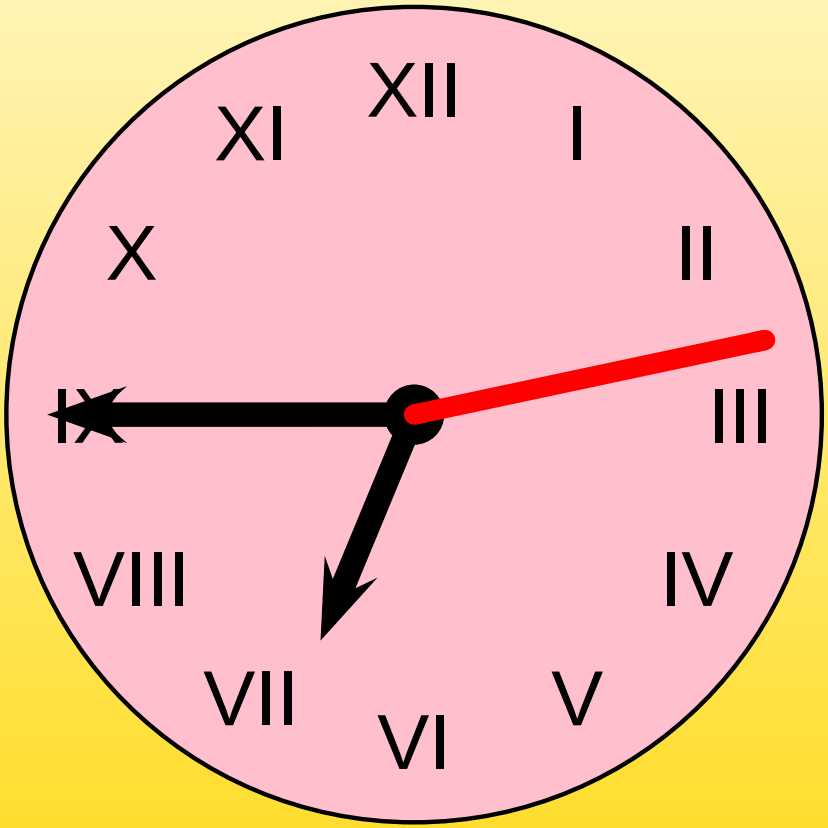


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 14s

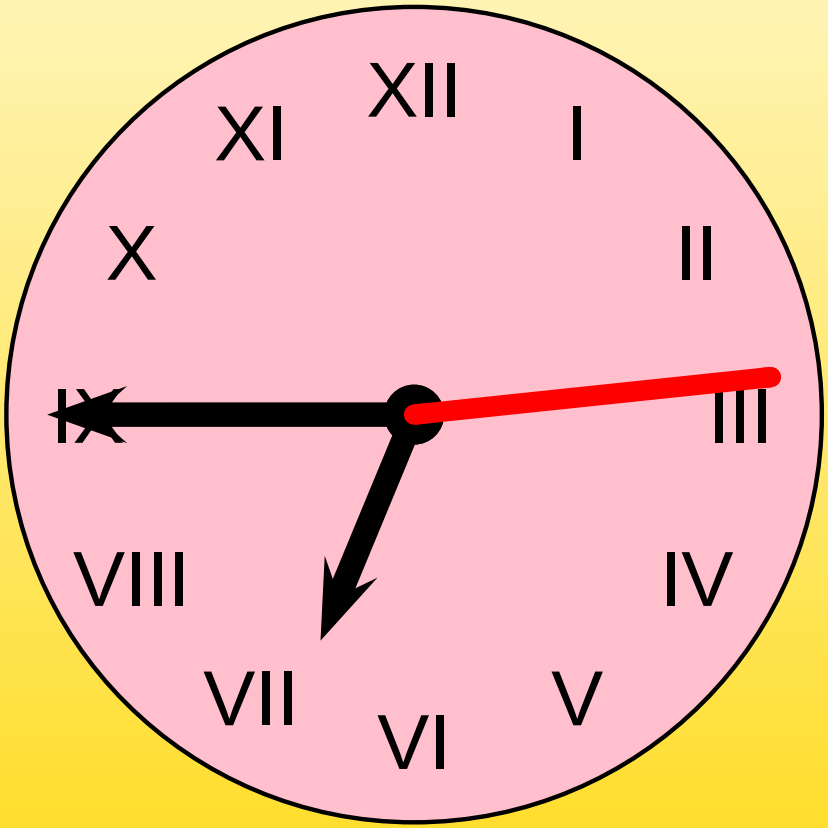


Figure 4: Clock with split-second hand

End of animation

6 – Clock with split-second hand

Document compiled at: 18h 45m 15s

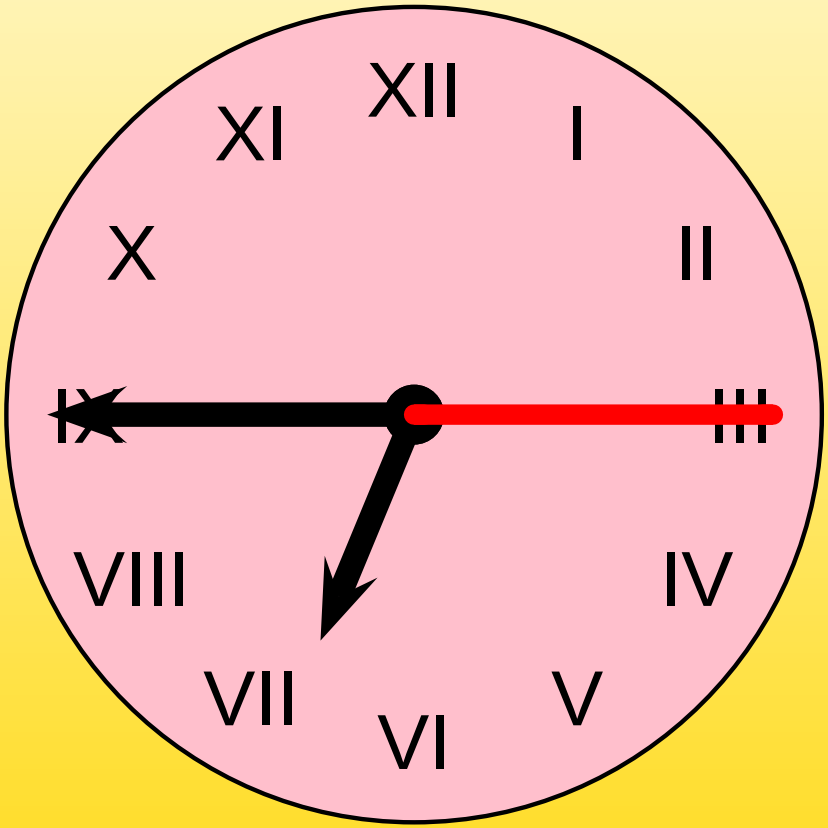


Figure 4: Clock with split-second hand

End of animation

7 – Random walk

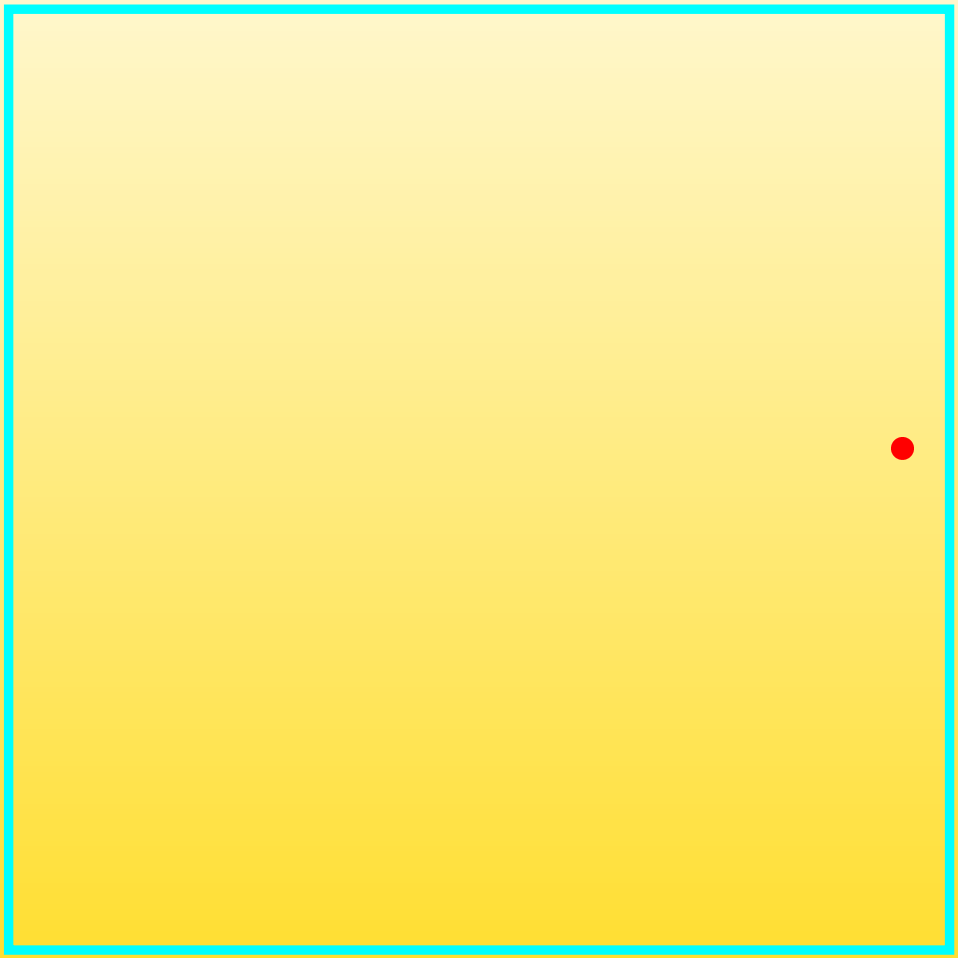


Figure 5: Random walk

End of animation

7 – Random walk

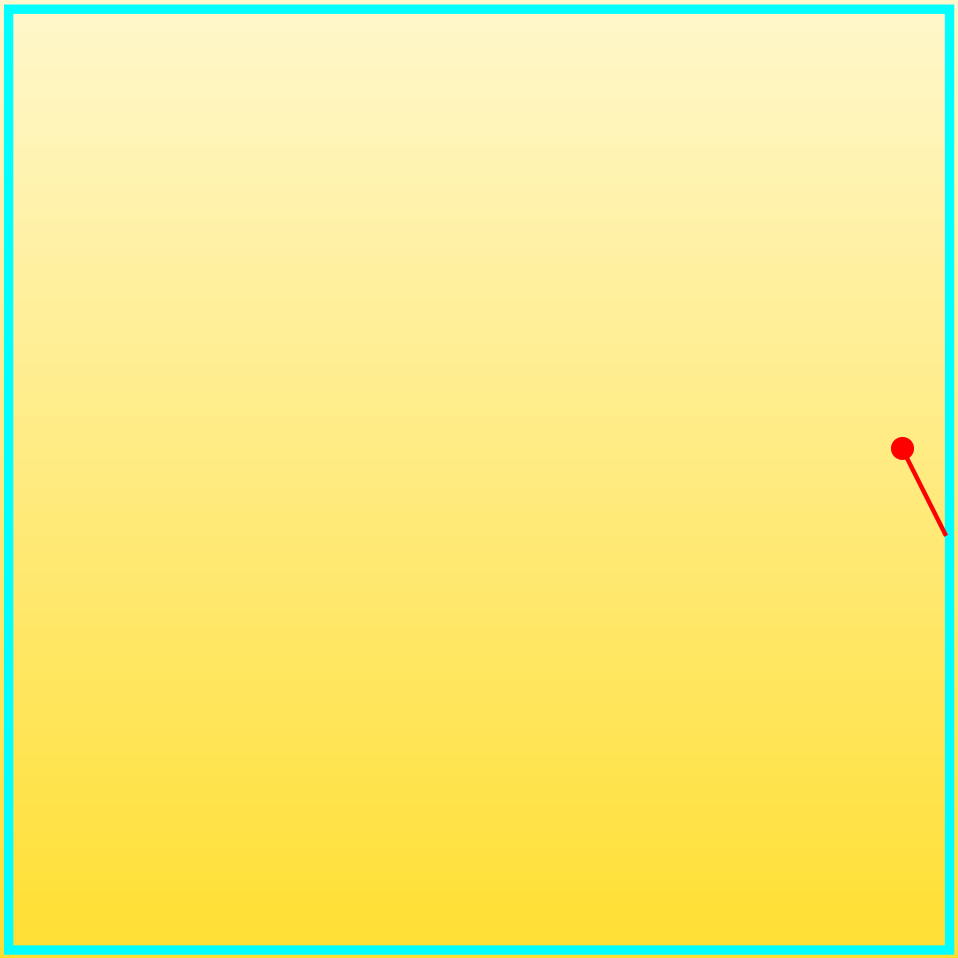


Figure 5: Random walk

End of animation

7 – Random walk

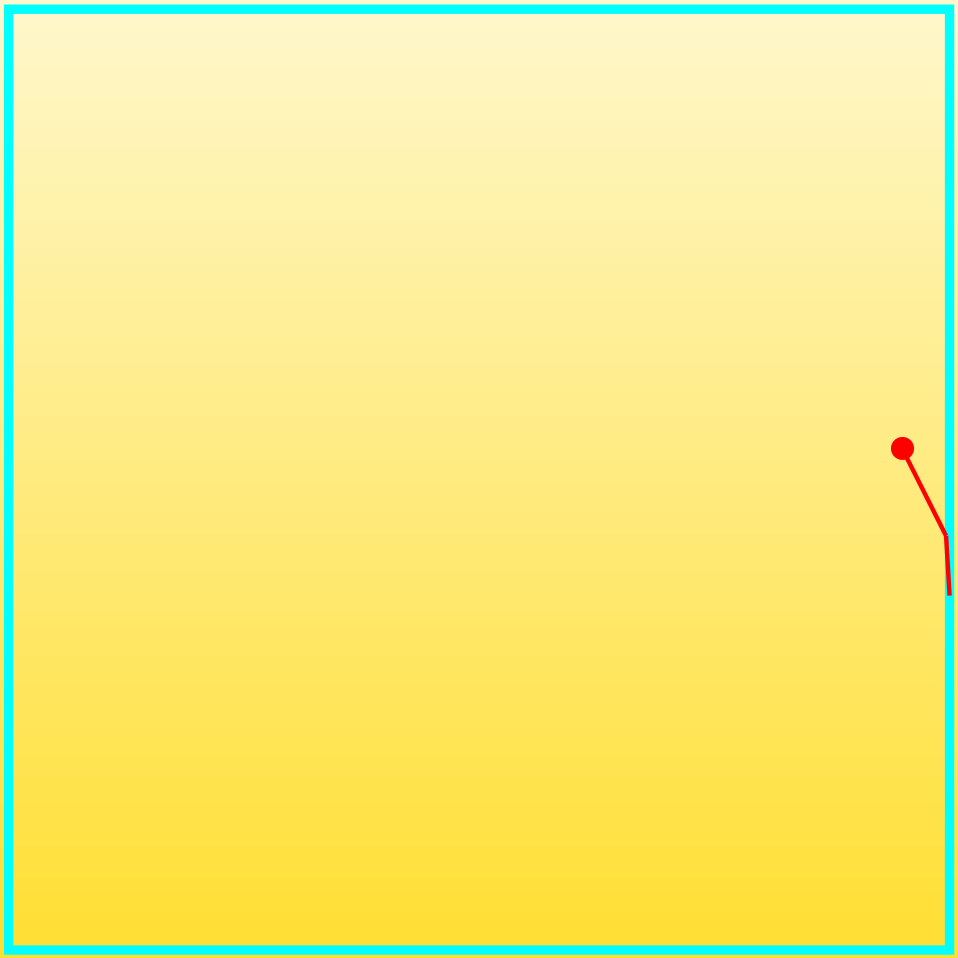


Figure 5: Random walk

End of animation

7 – Random walk

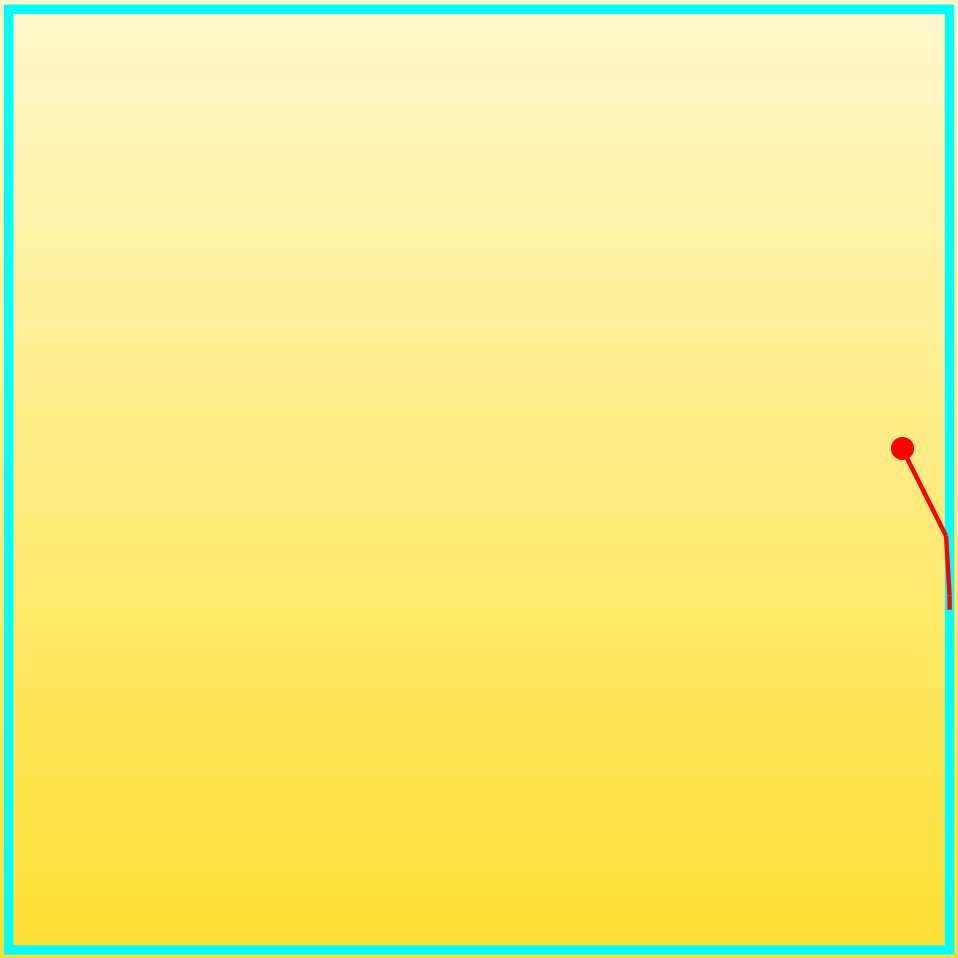


Figure 5: Random walk

End of animation

7 – Random walk

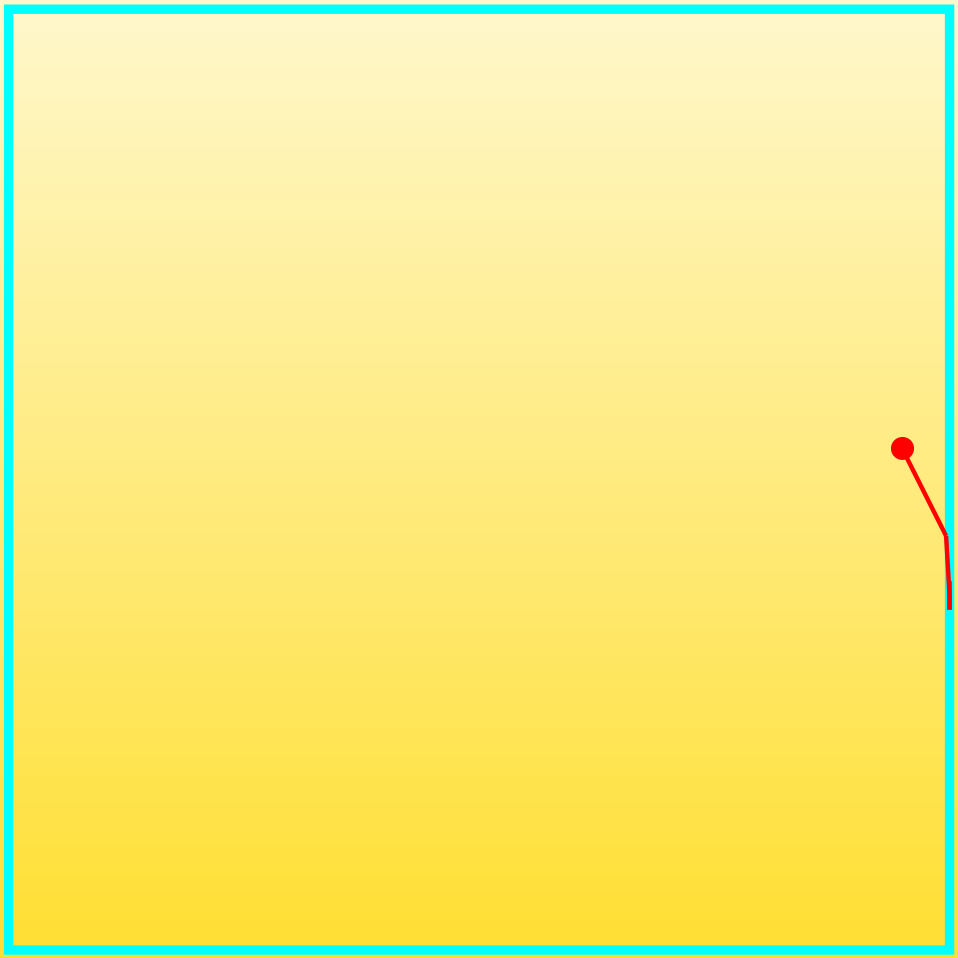


Figure 5: Random walk

End of animation

7 – Random walk

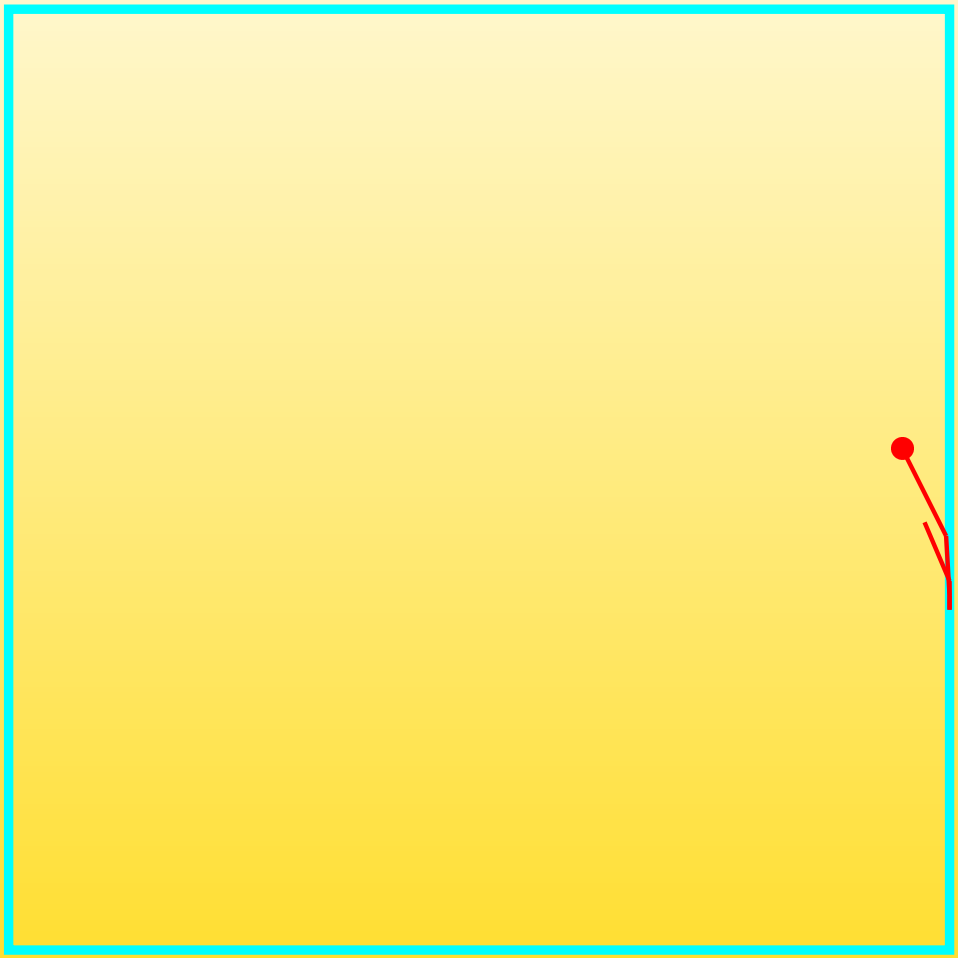


Figure 5: Random walk

End of animation

7 – Random walk

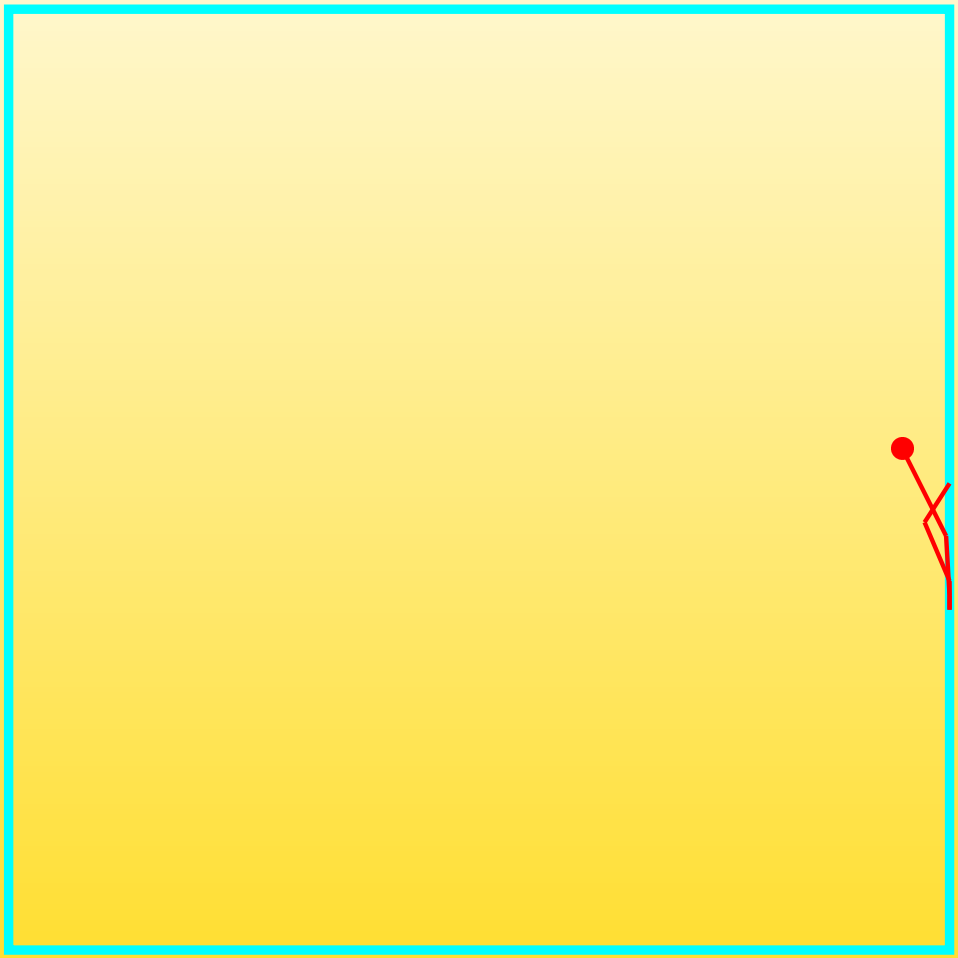


Figure 5: Random walk

End of animation

7 – Random walk

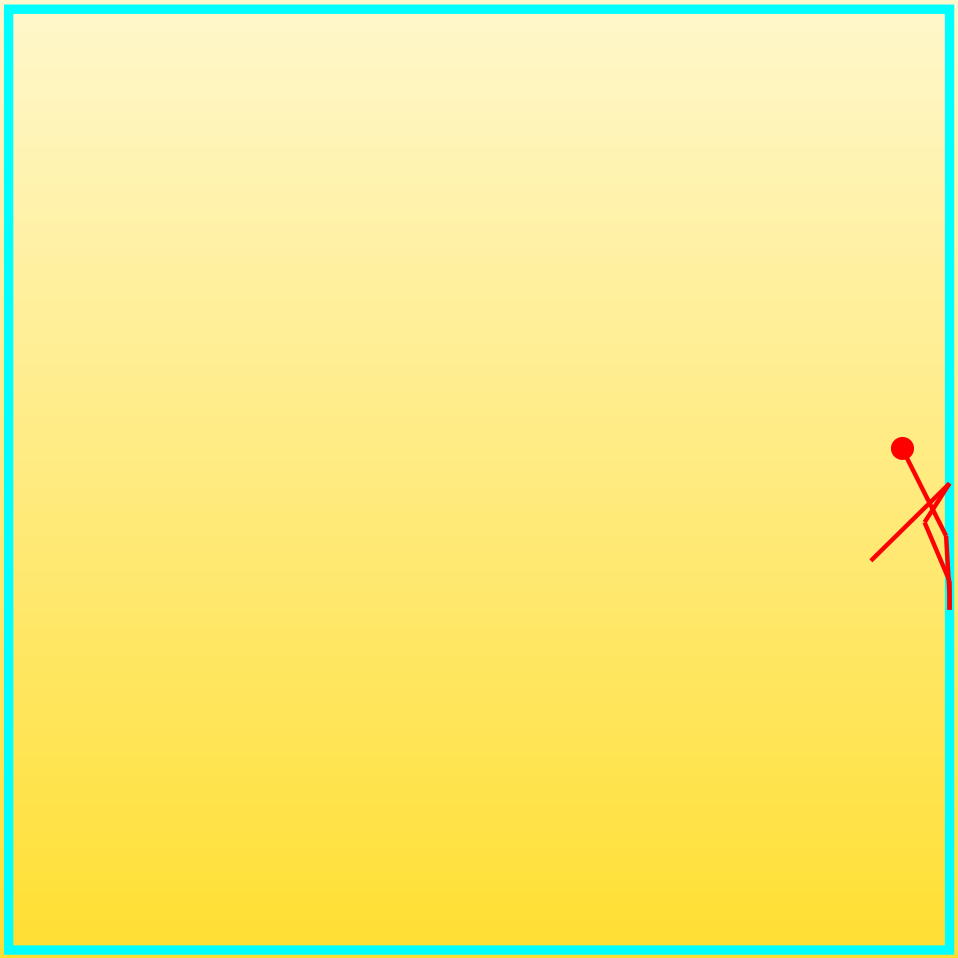


Figure 5: Random walk

End of animation

7 – Random walk

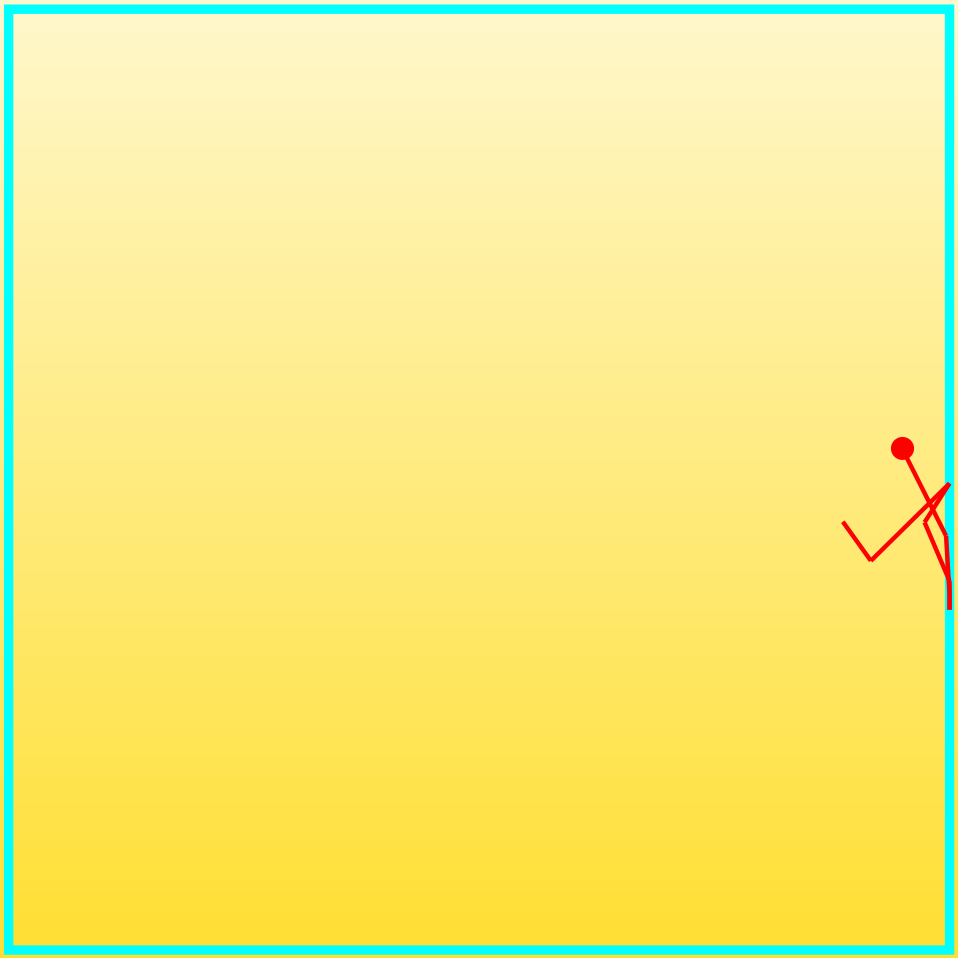


Figure 5: Random walk

End of animation

7 – Random walk

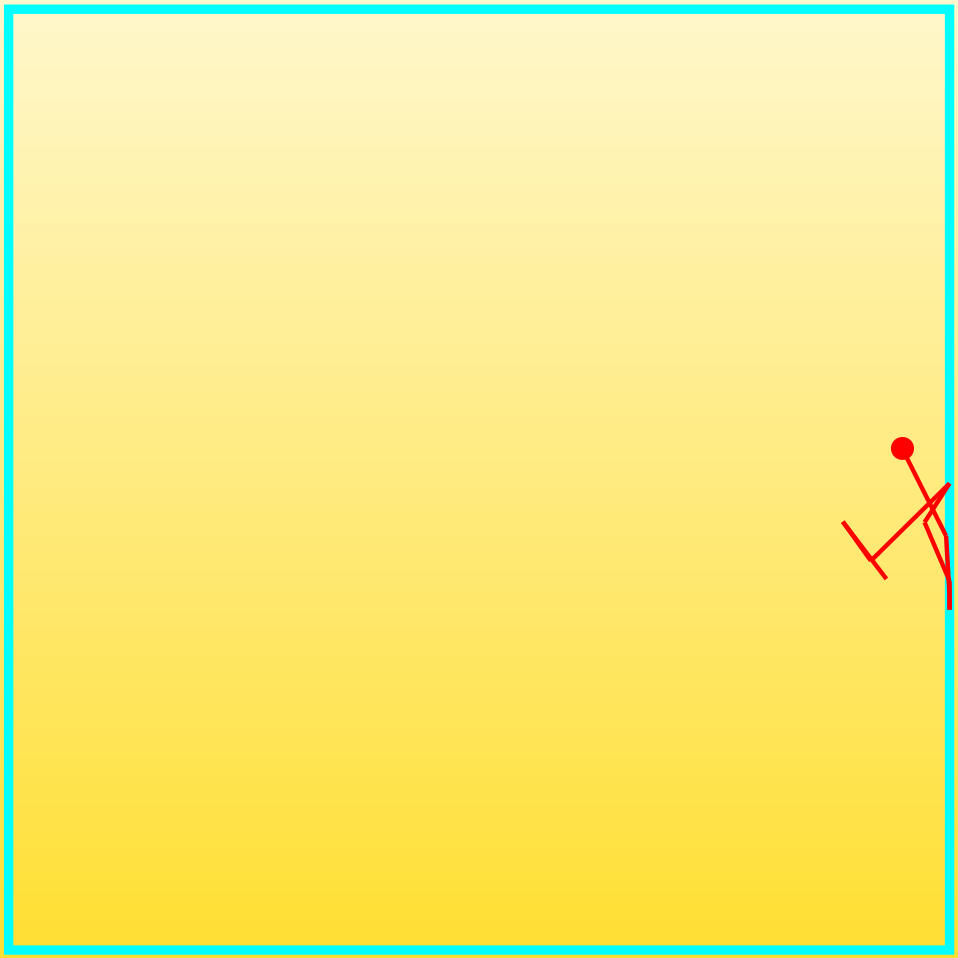


Figure 5: Random walk

End of animation

7 – Random walk

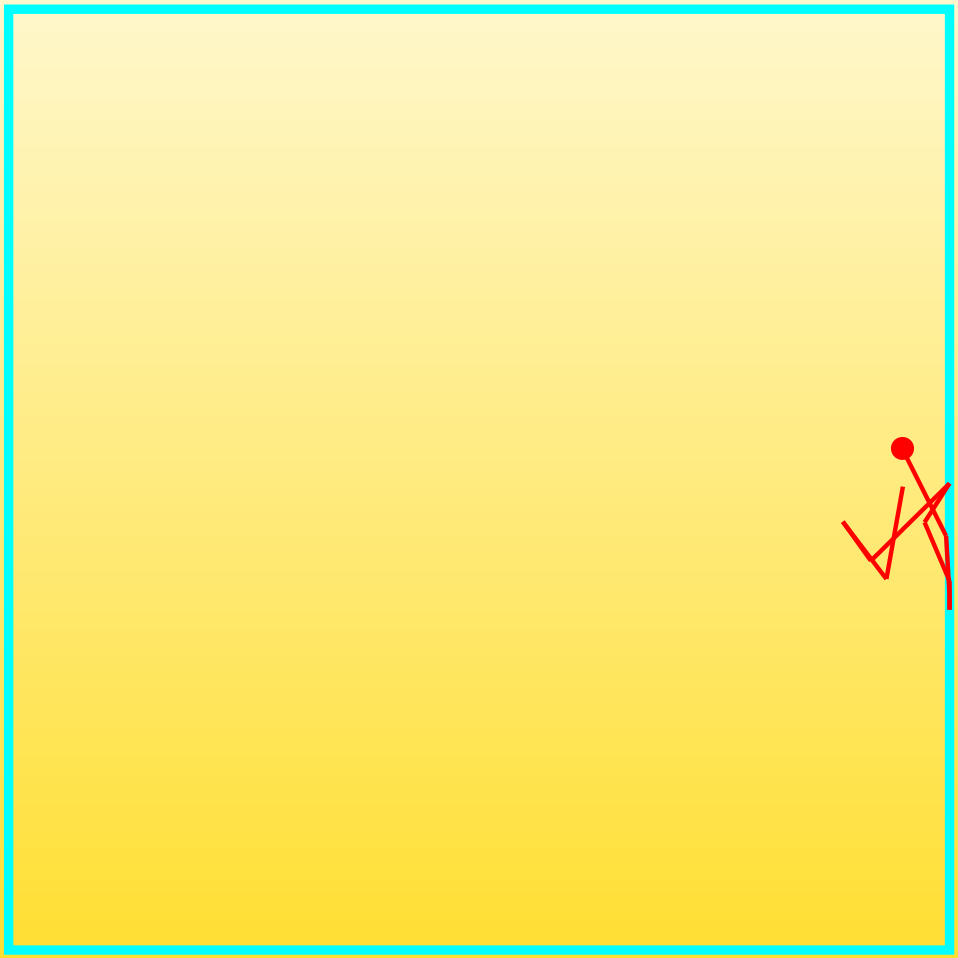


Figure 5: Random walk

End of animation

7 – Random walk

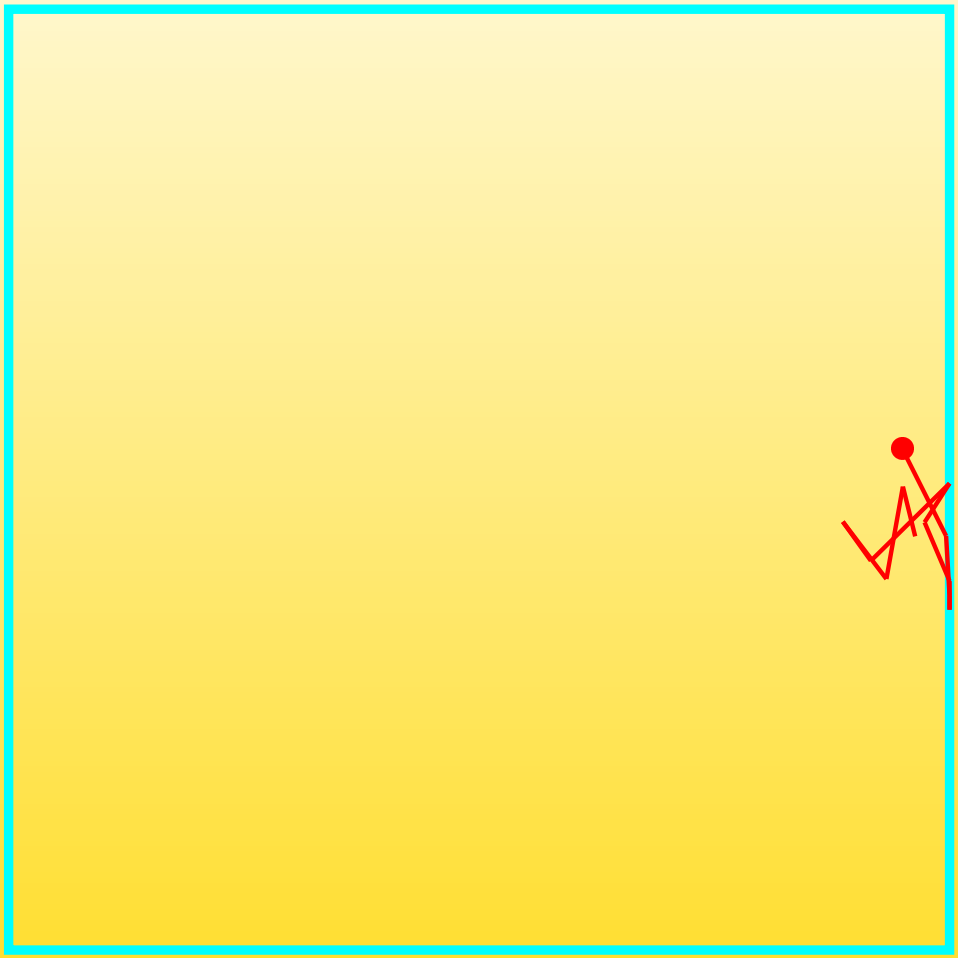


Figure 5: Random walk

End of animation

7 – Random walk

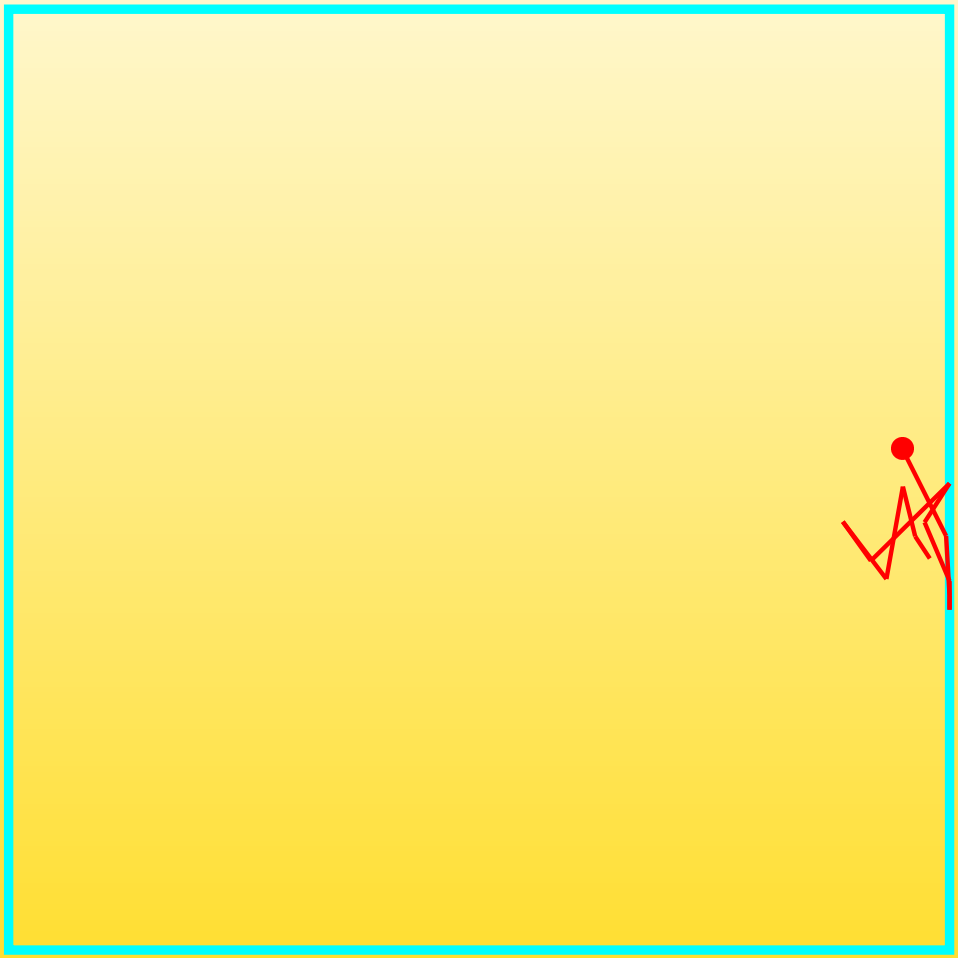


Figure 5: Random walk

End of animation

7 – Random walk

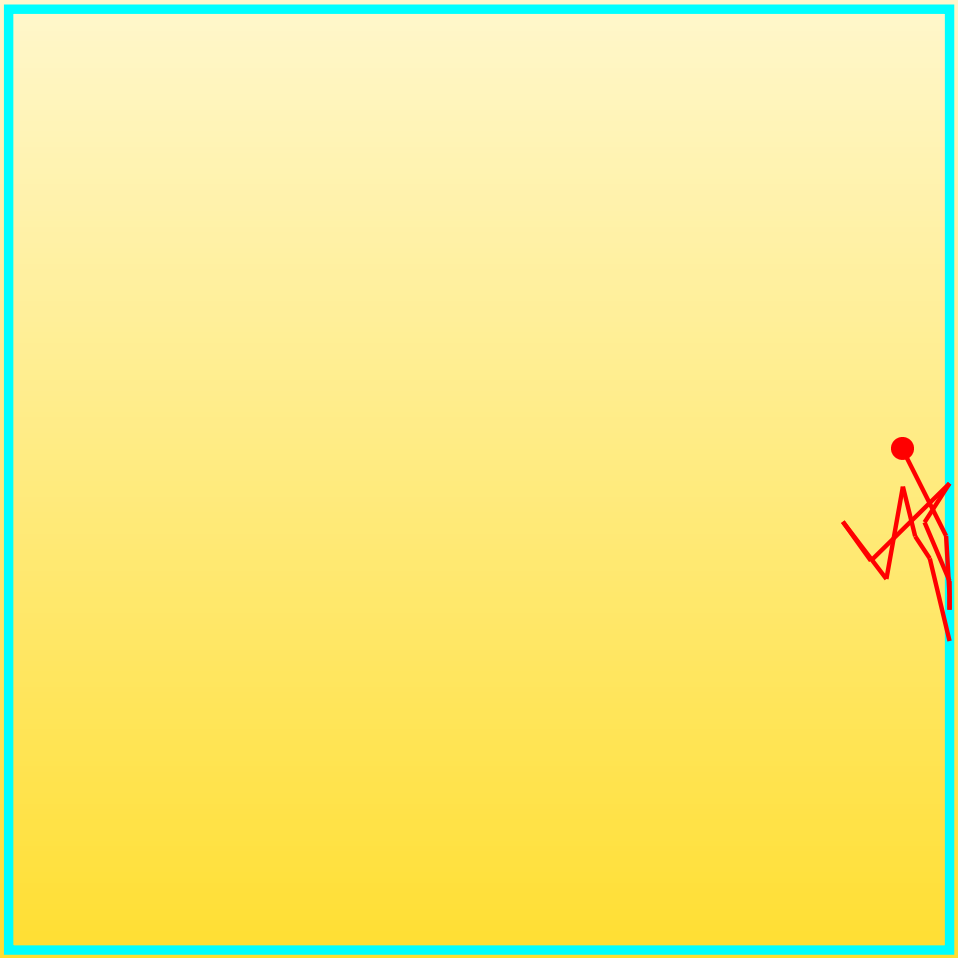


Figure 5: Random walk

End of animation

7 – Random walk

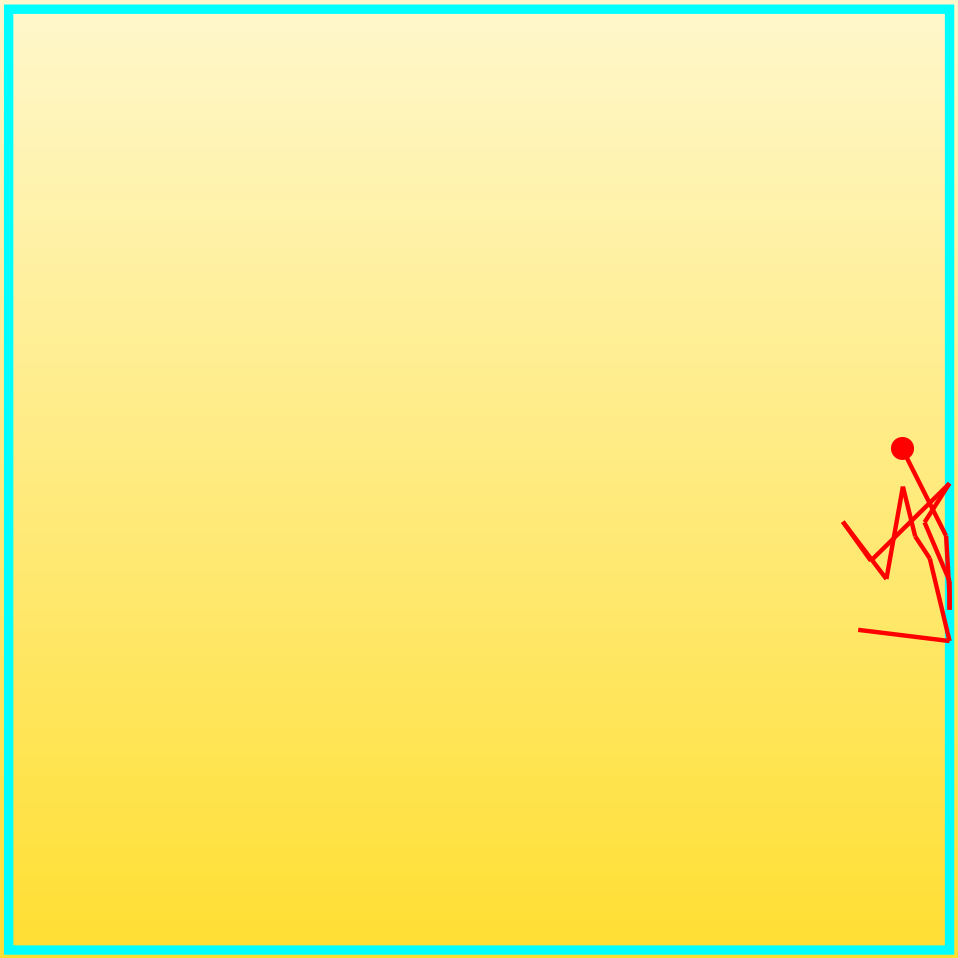


Figure 5: Random walk

End of animation

7 – Random walk

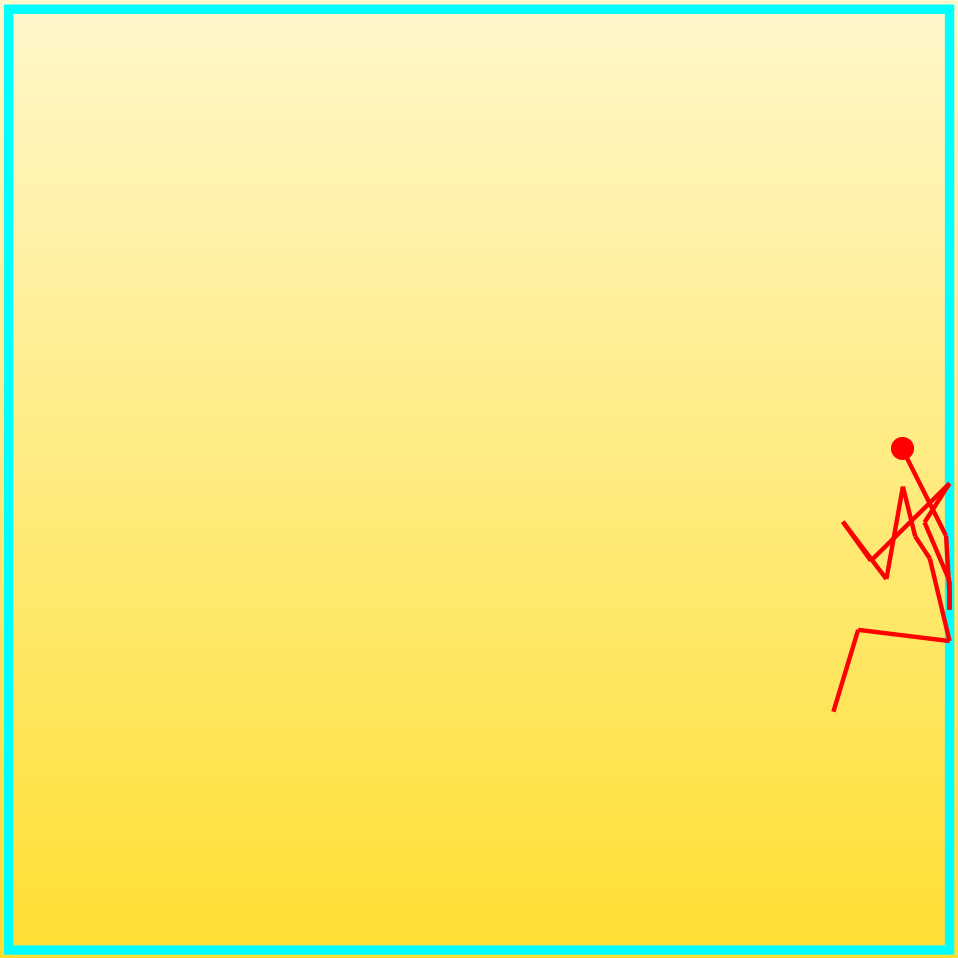


Figure 5: Random walk

End of animation

7 – Random walk

Figure 5: Random walk

End of animation

7 – Random walk

Figure 5: Random walk

End of animation

7 – Random walk

Figure 5: Random walk

End of animation

7 – Random walk

Figure 5: Random walk

End of animation

7 – Random walk

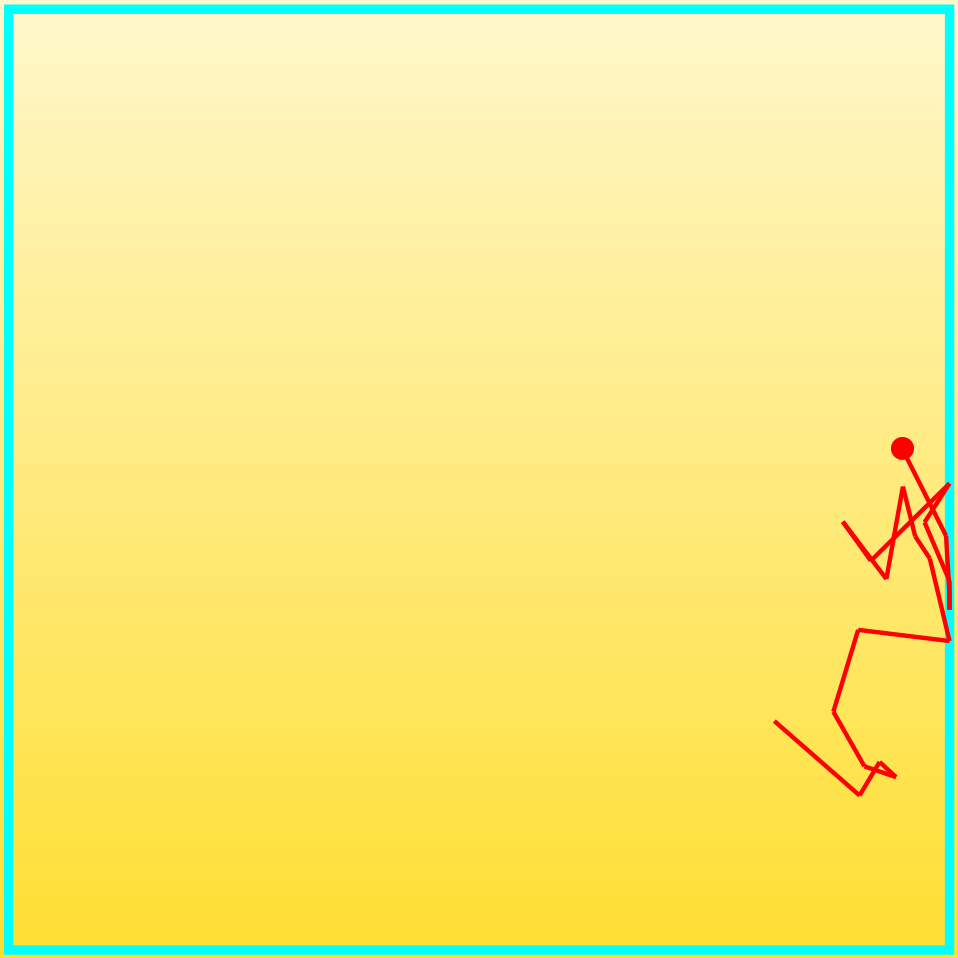


Figure 5: Random walk

End of animation

7 – Random walk

Figure 5: Random walk

End of animation

7 – Random walk

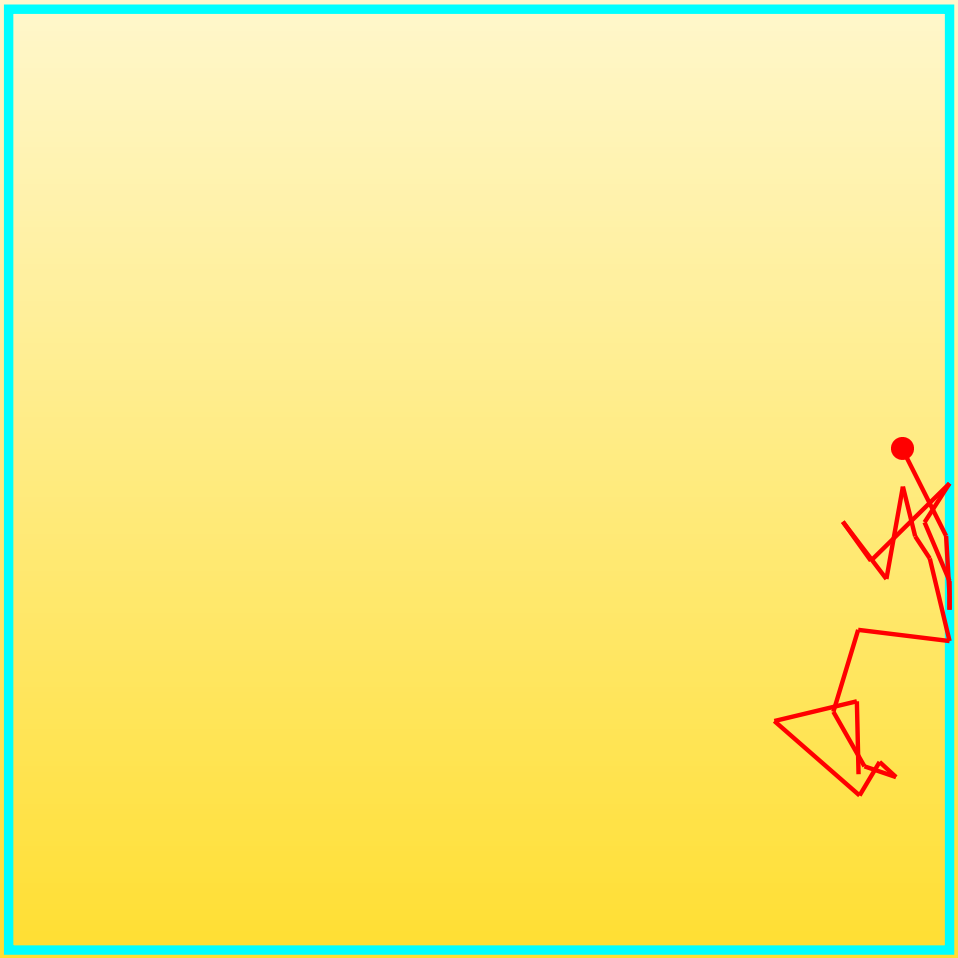


Figure 5: Random walk

End of animation

7 – Random walk

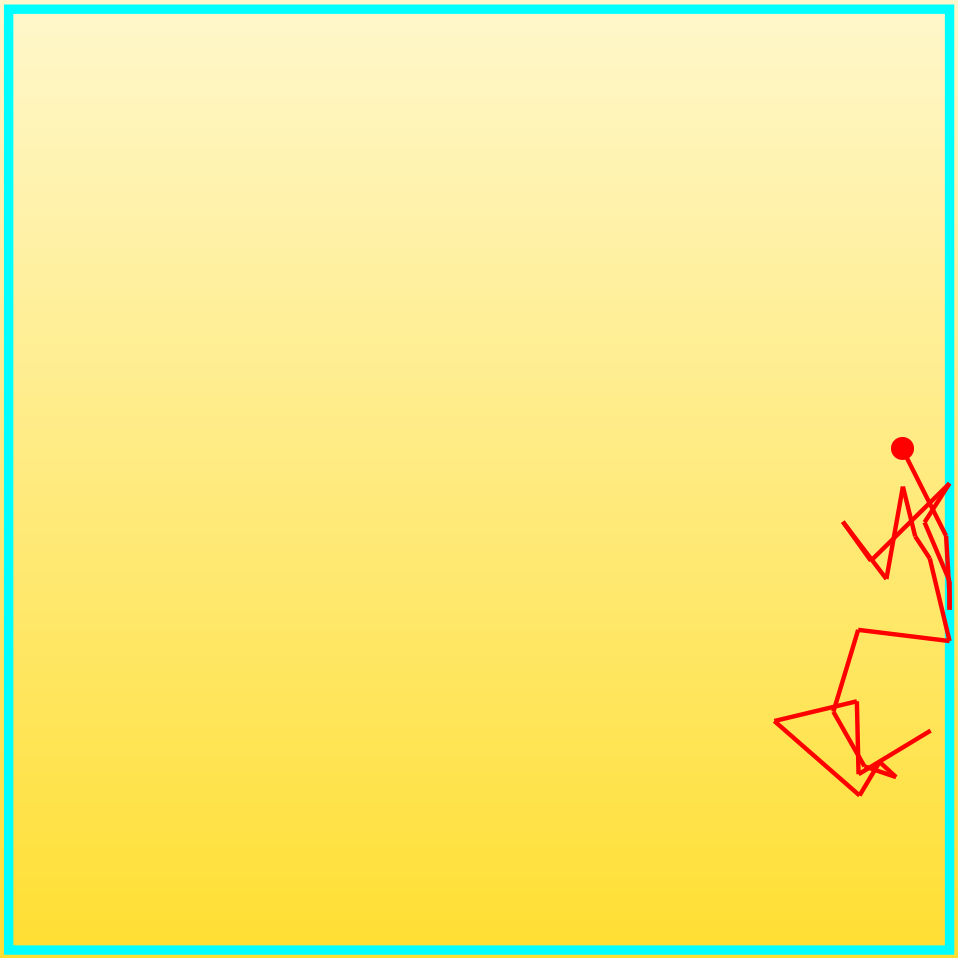


Figure 5: Random walk

End of animation

7 – Random walk

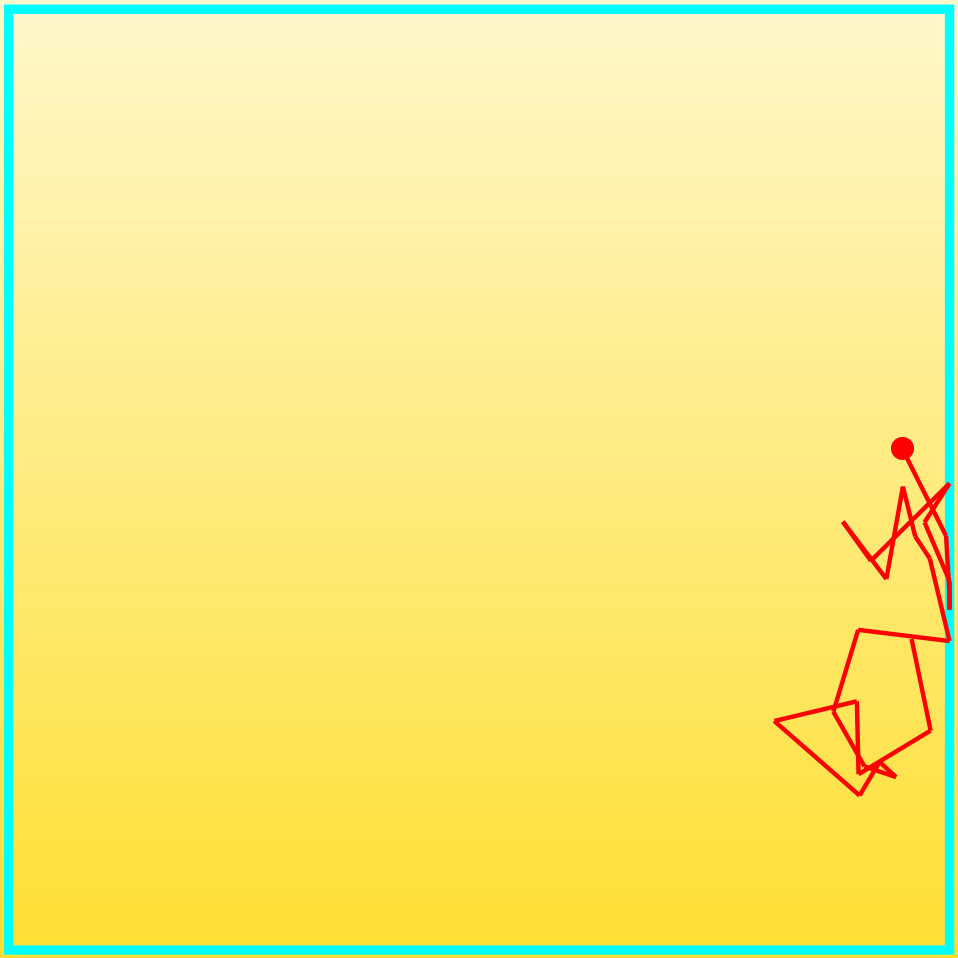


Figure 5: Random walk

End of animation

7 – Random walk

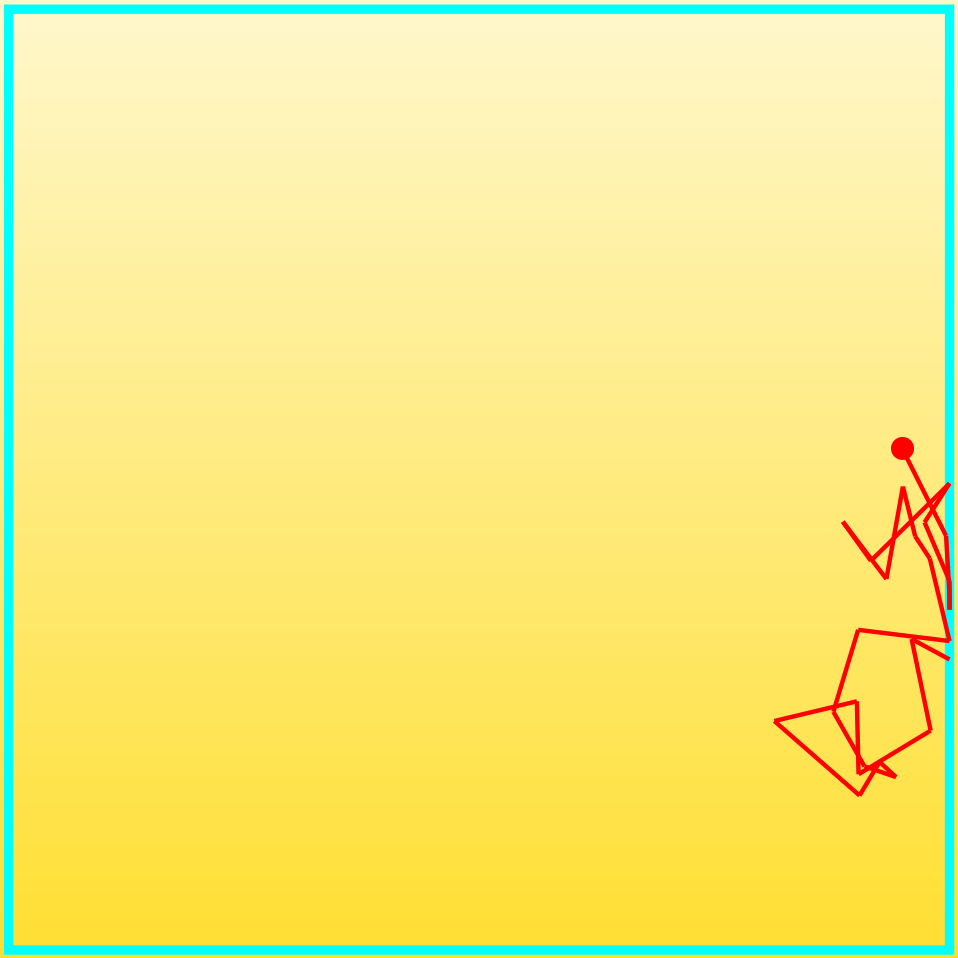


Figure 5: Random walk

End of animation

7 – Random walk

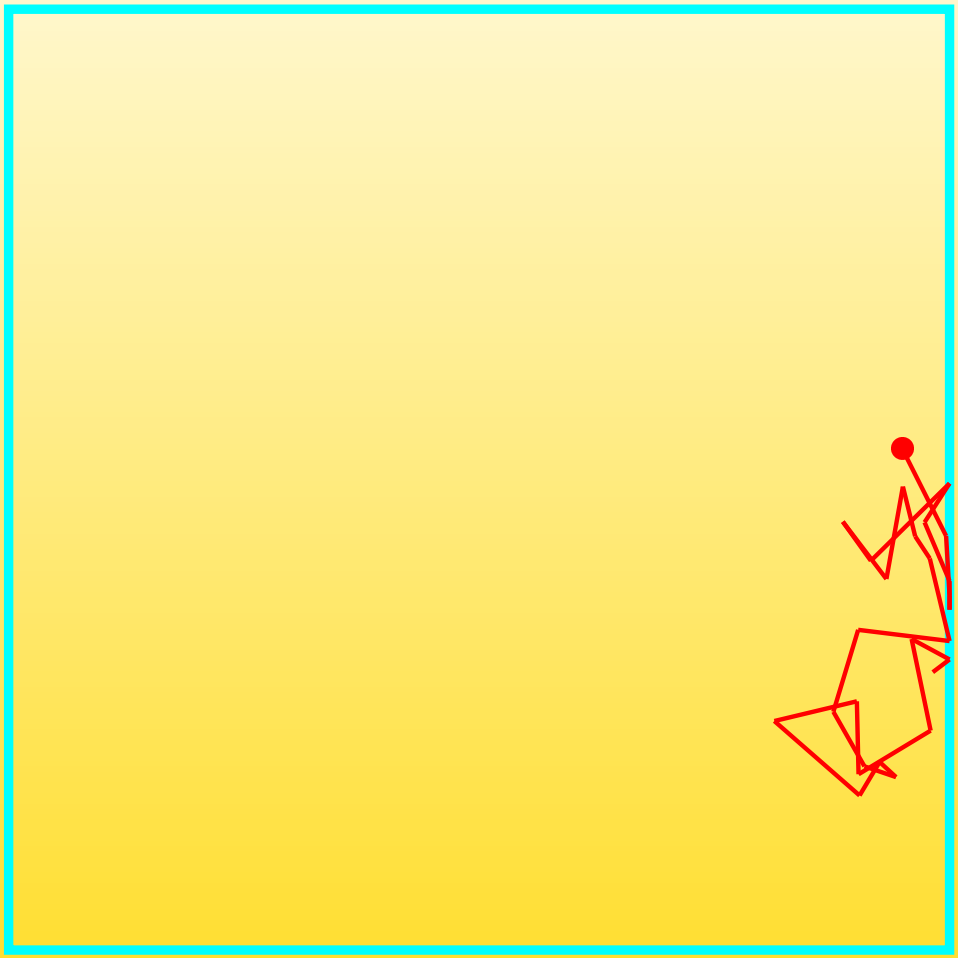


Figure 5: Random walk

End of animation

7 – Random walk

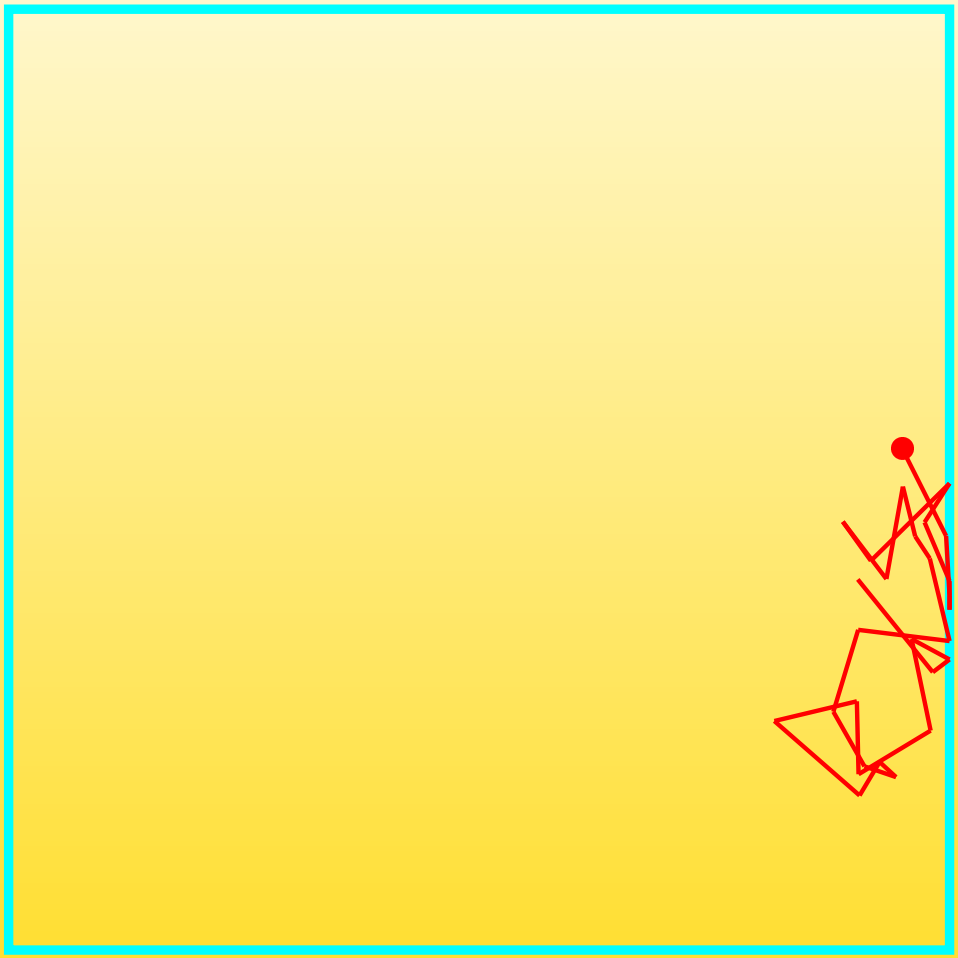


Figure 5: Random walk

End of animation

7 – Random walk

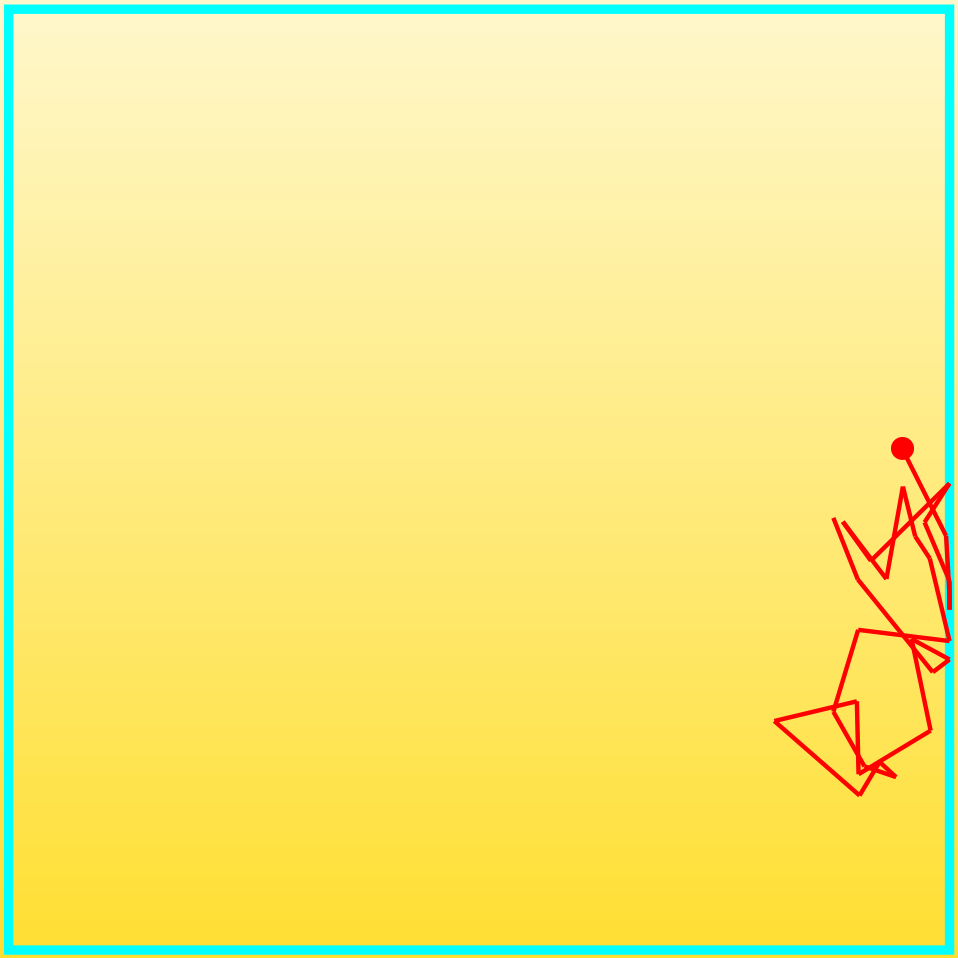


Figure 5: Random walk

End of animation

7 – Random walk

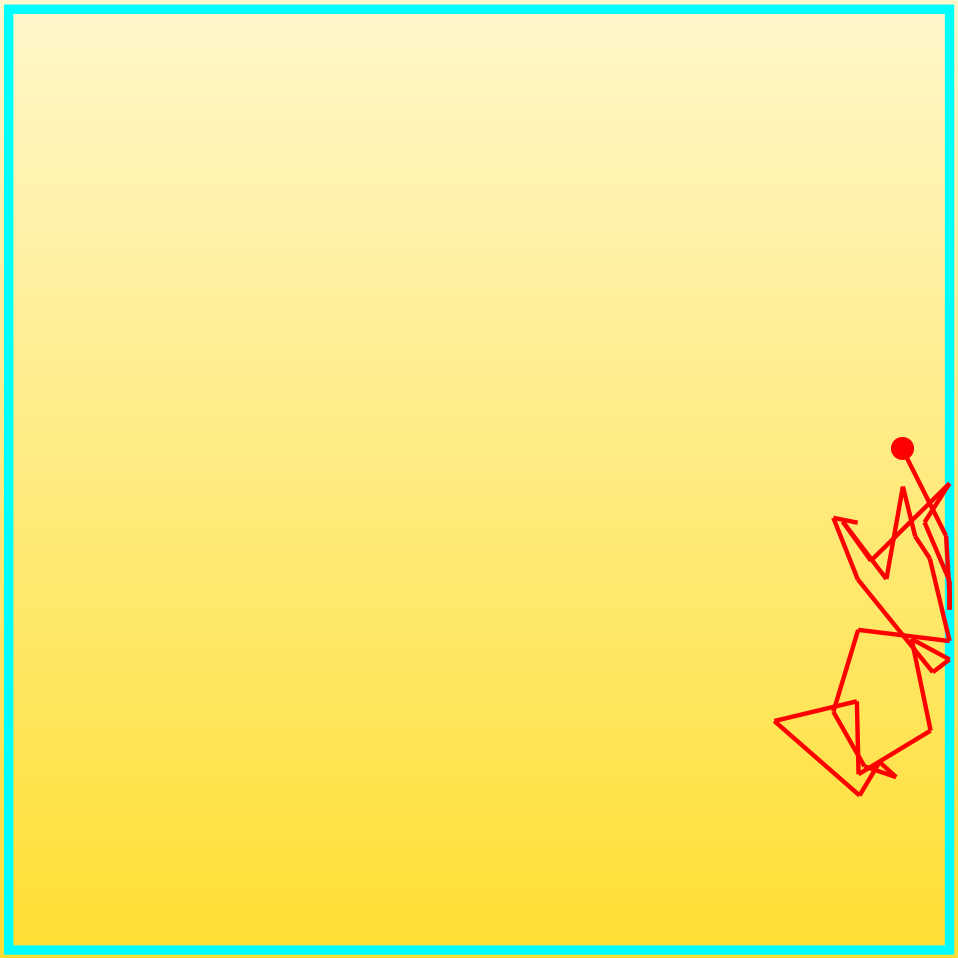


Figure 5: Random walk

End of animation

7 – Random walk

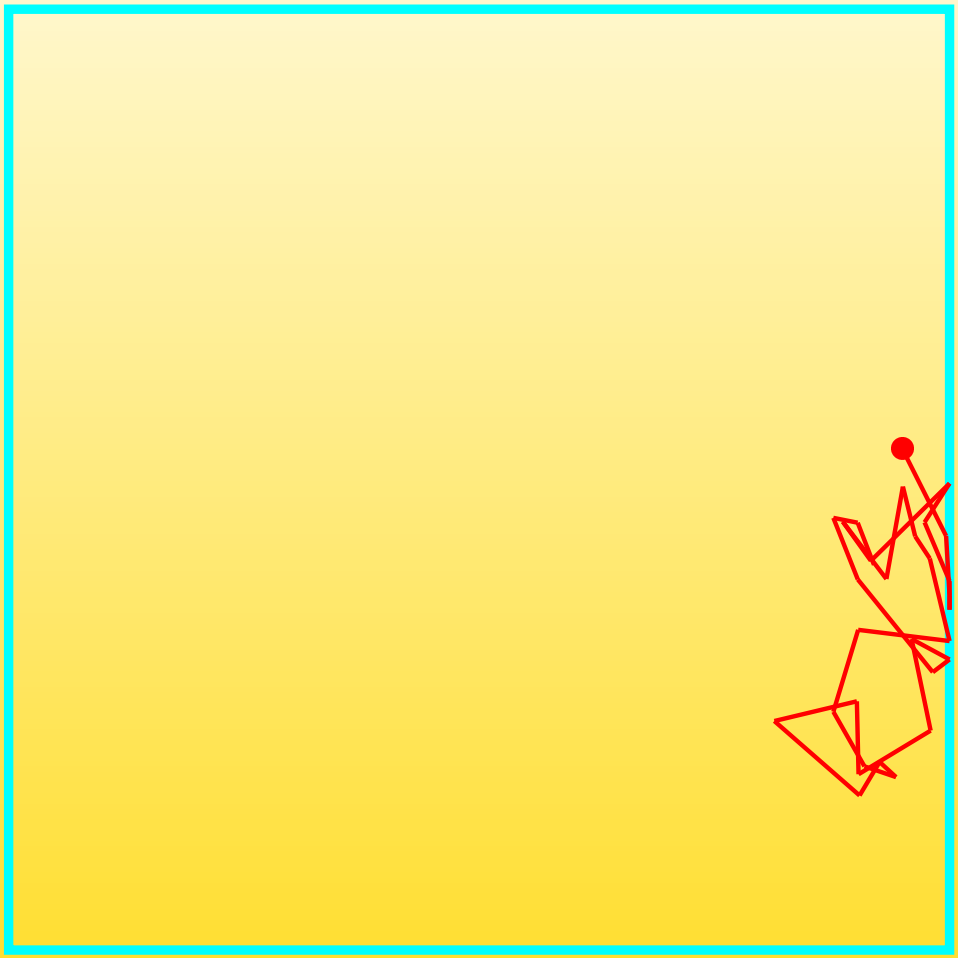


Figure 5: Random walk

End of animation

7 – Random walk

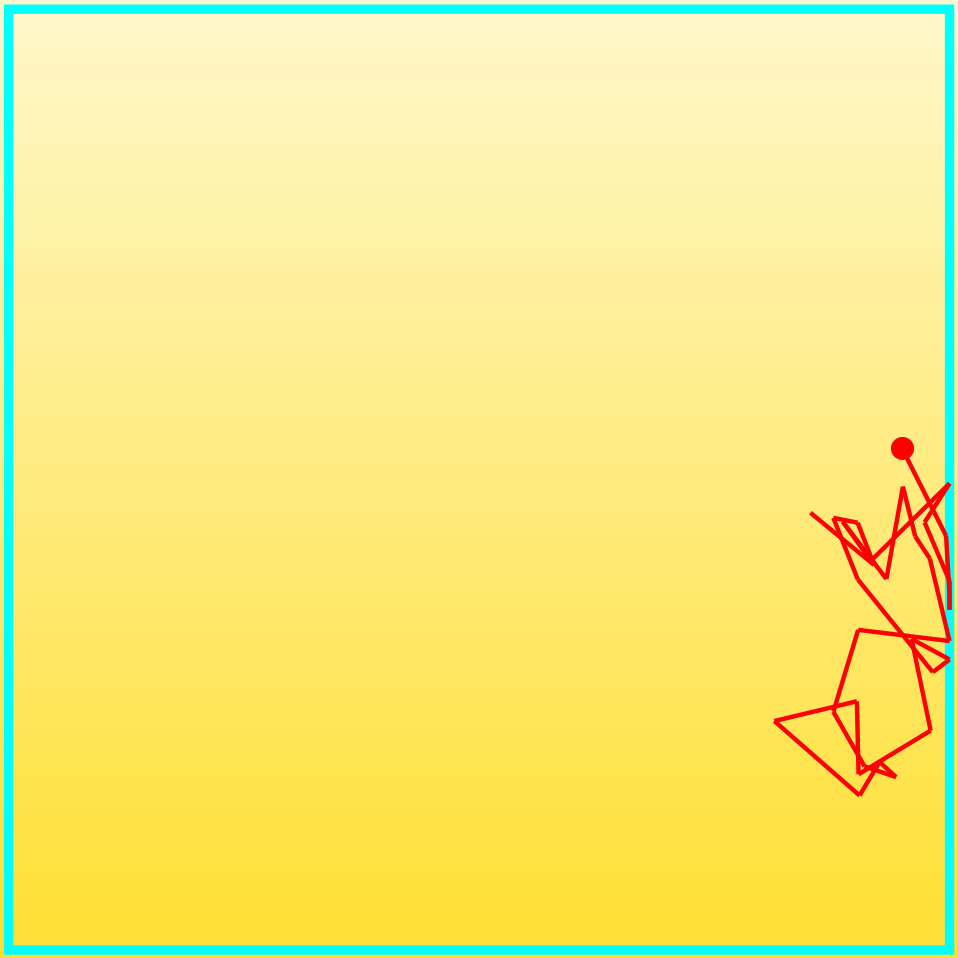


Figure 5: Random walk

End of animation

7 – Random walk

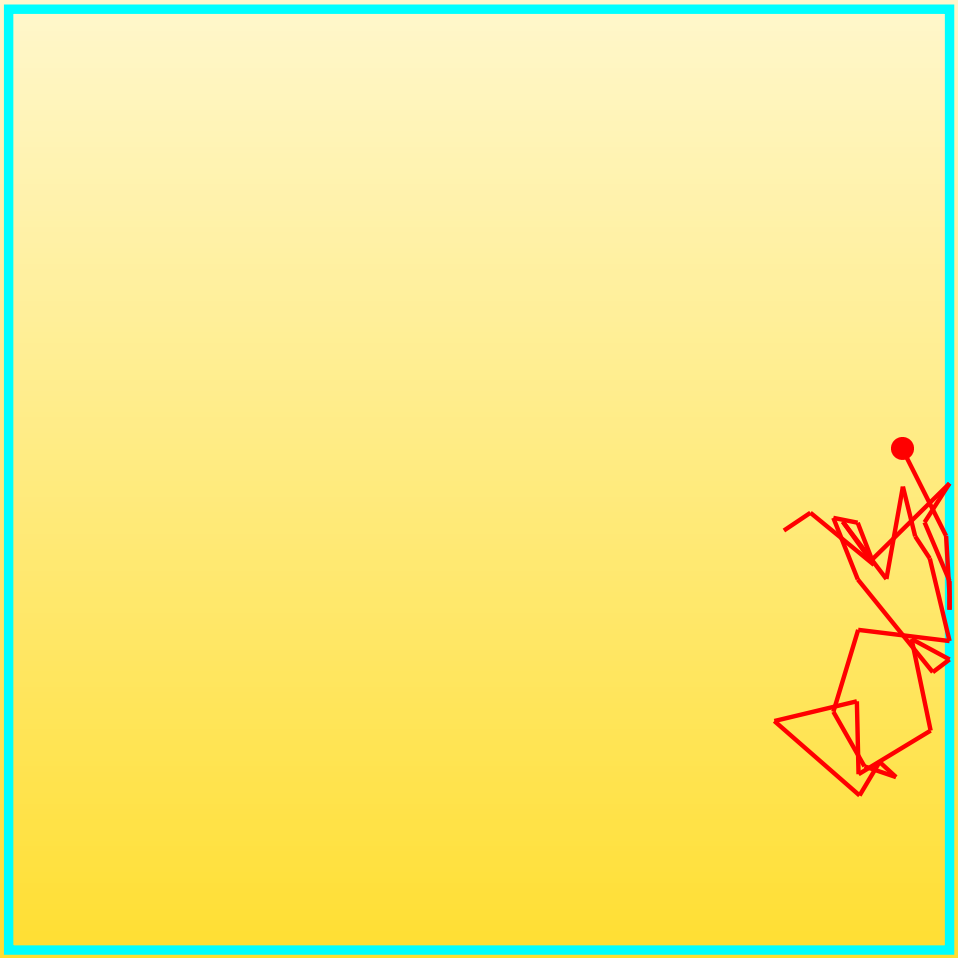


Figure 5: Random walk

End of animation

7 – Random walk

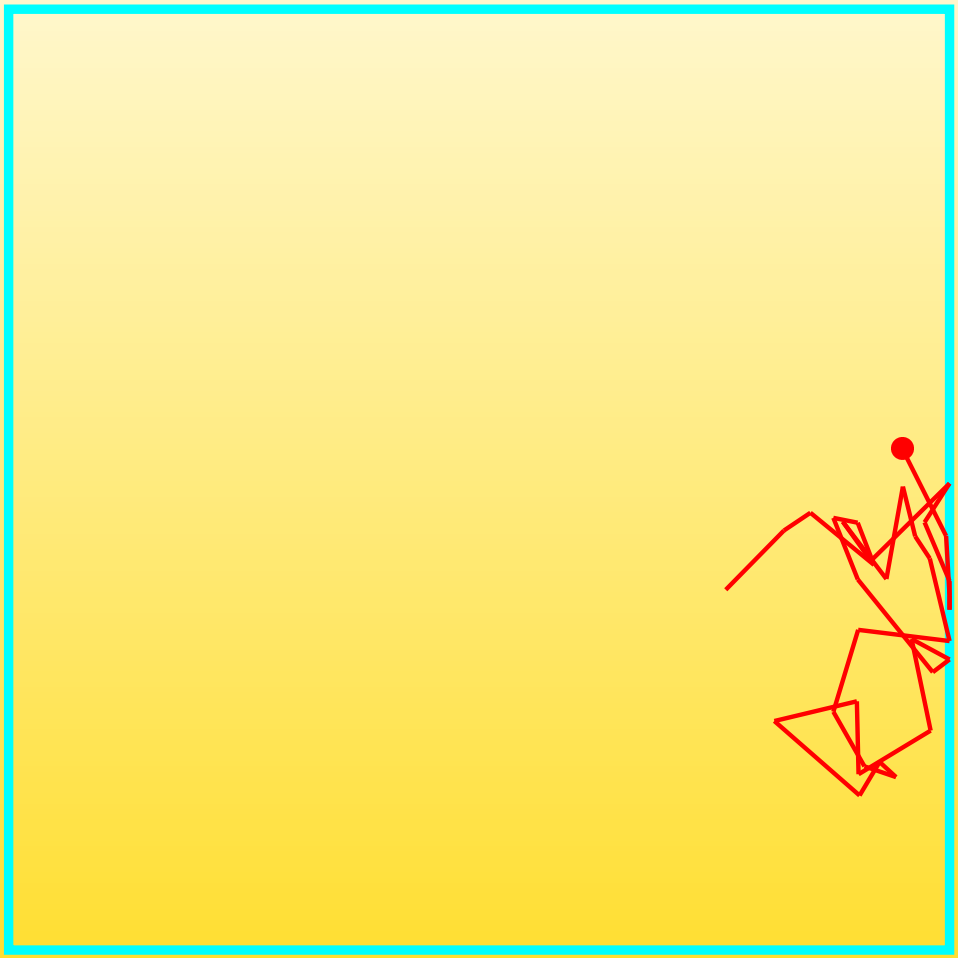


Figure 5: Random walk

End of animation

7 – Random walk

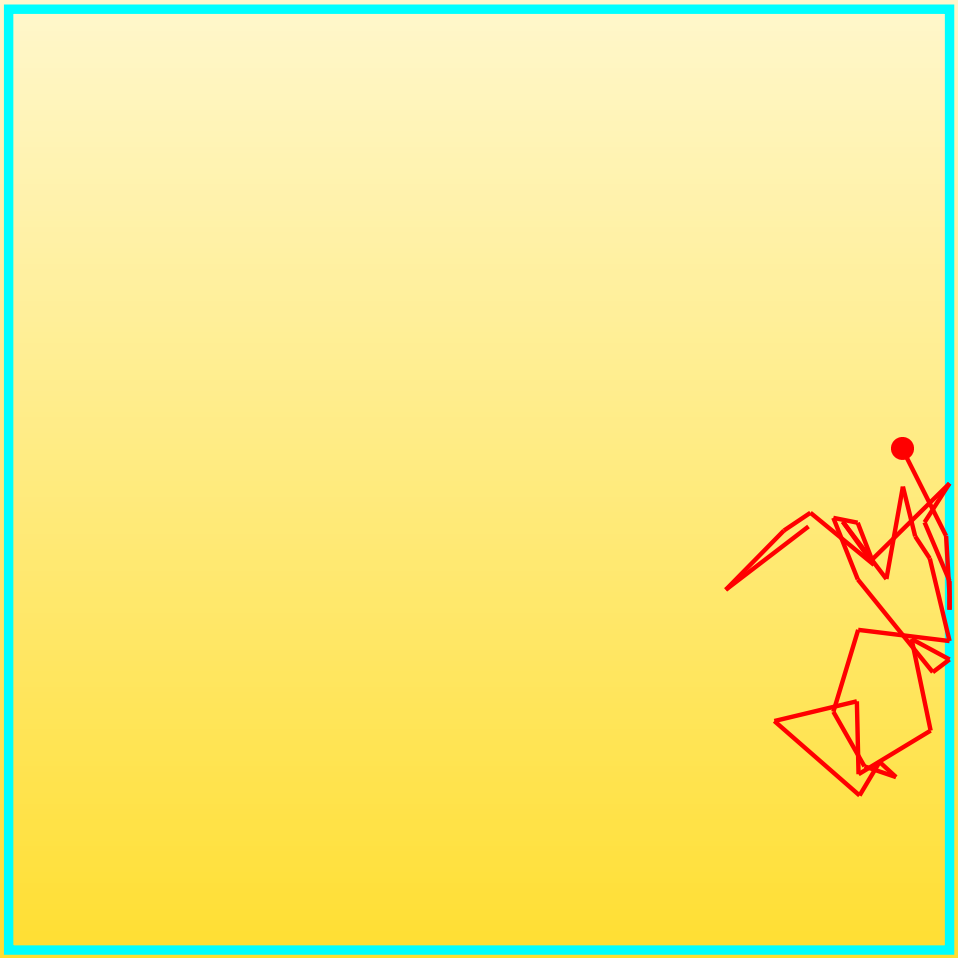


Figure 5: Random walk

End of animation

7 – Random walk

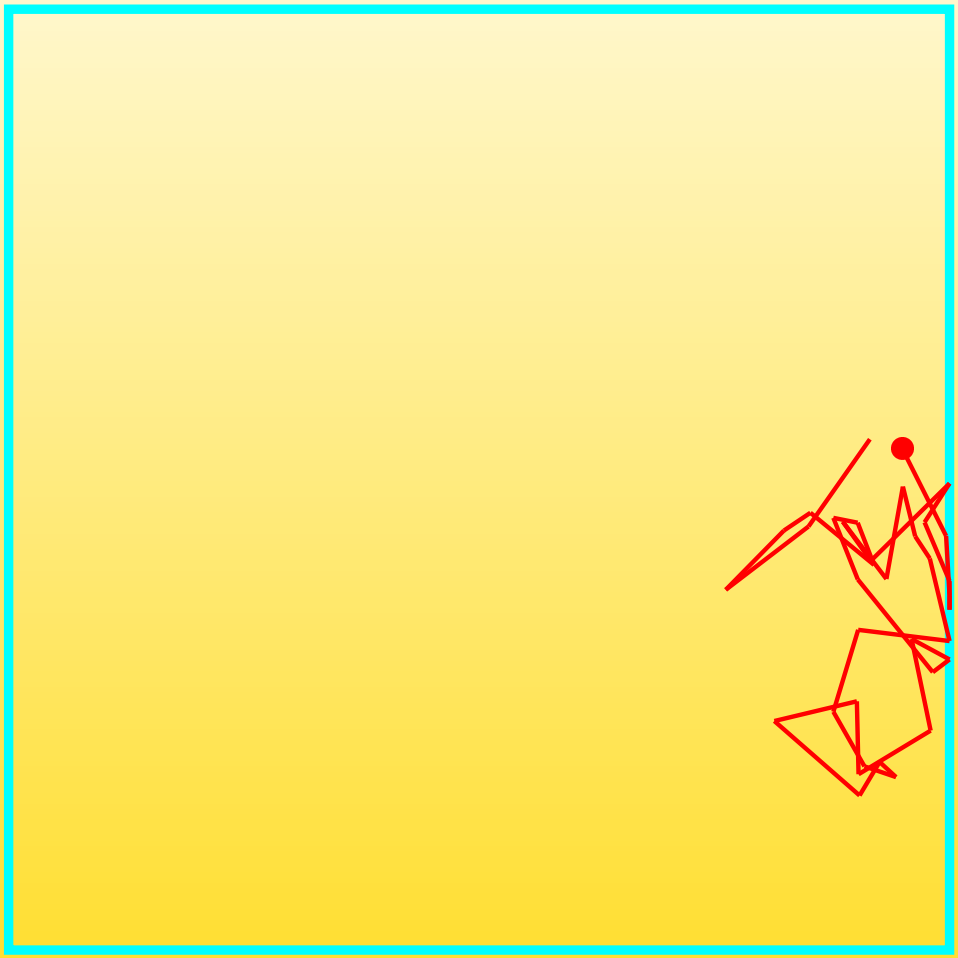


Figure 5: Random walk

End of animation

7 – Random walk

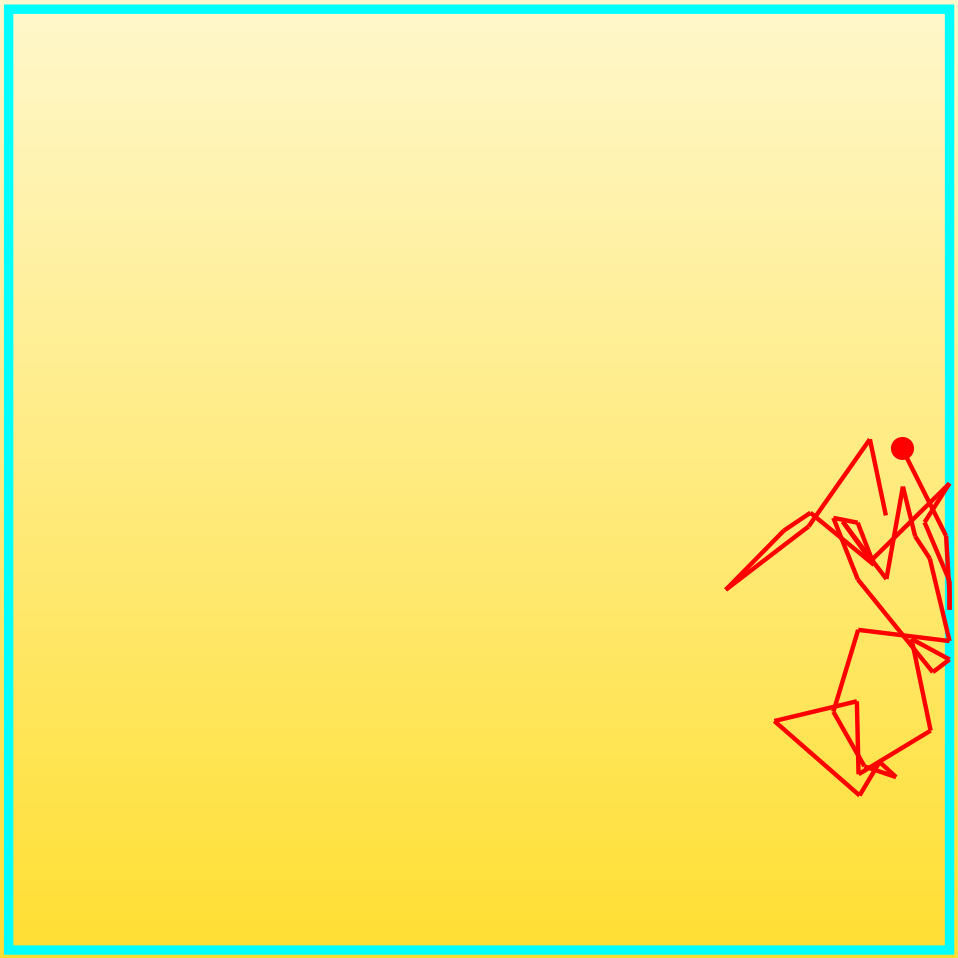


Figure 5: Random walk

End of animation

7 – Random walk

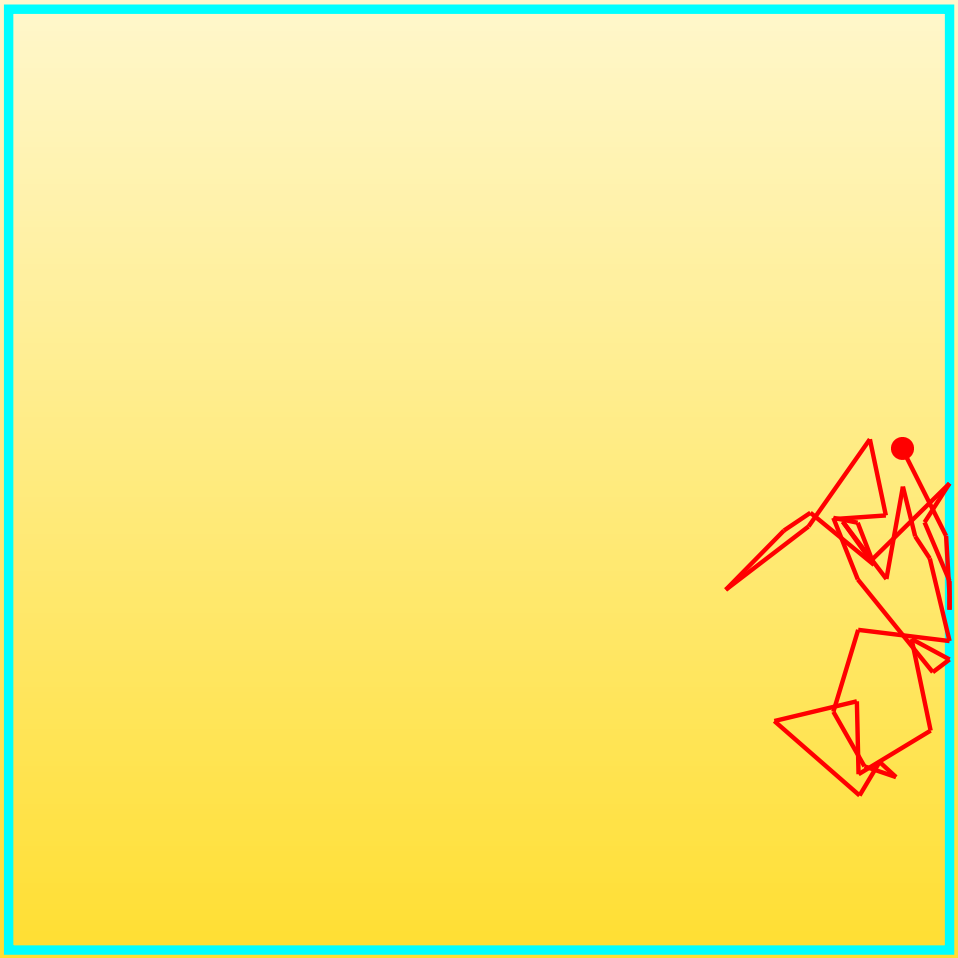


Figure 5: Random walk

End of animation

7 – Random walk

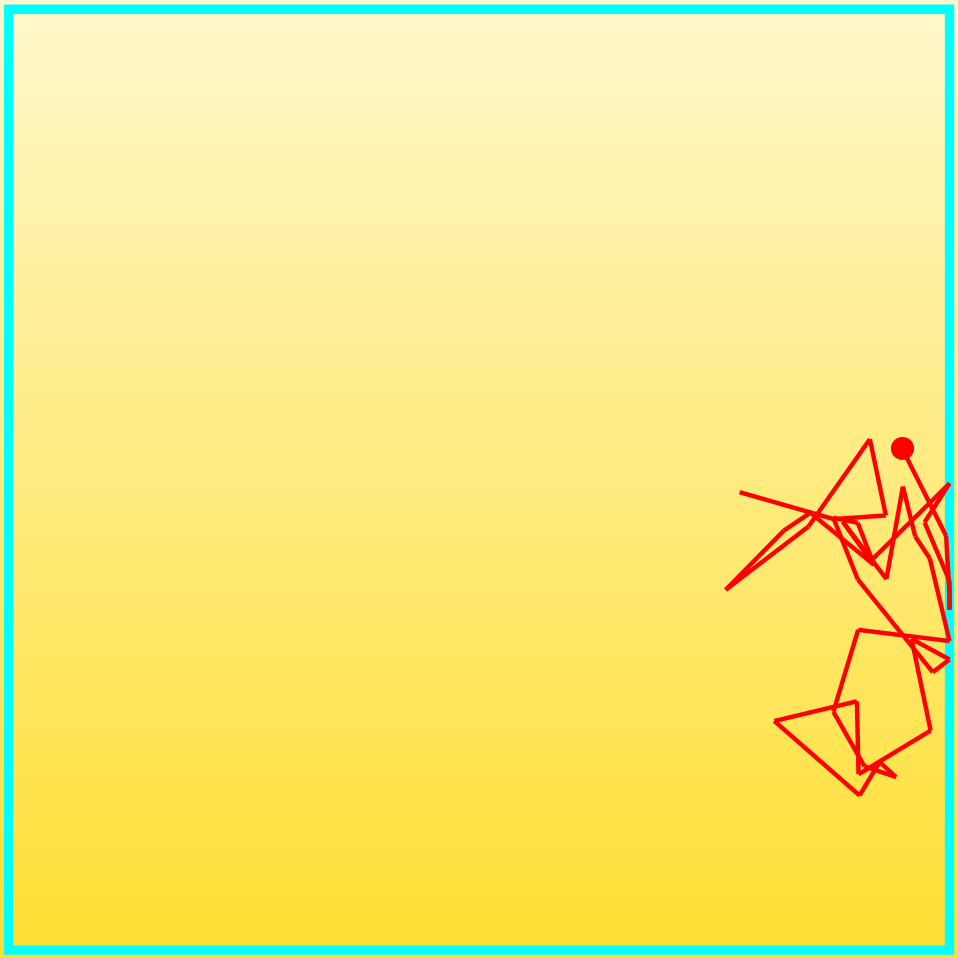


Figure 5: Random walk

End of animation

7 – Random walk

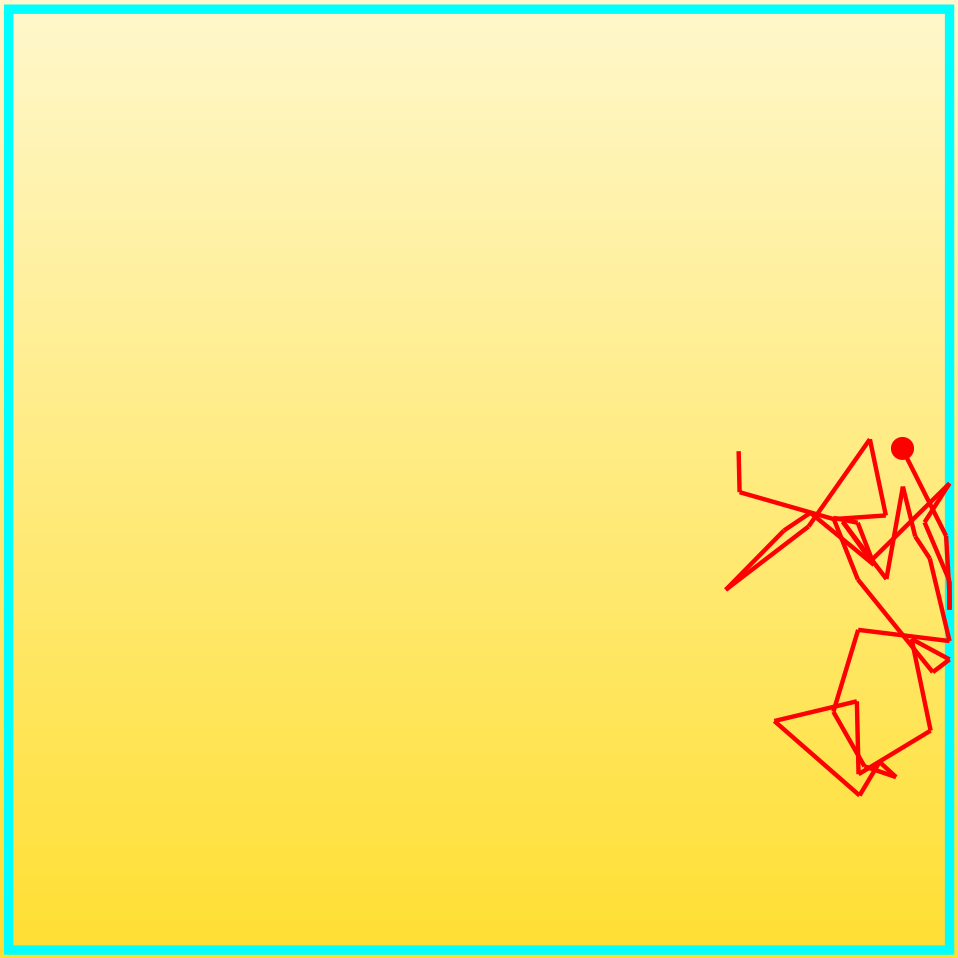


Figure 5: Random walk

End of animation

7 – Random walk

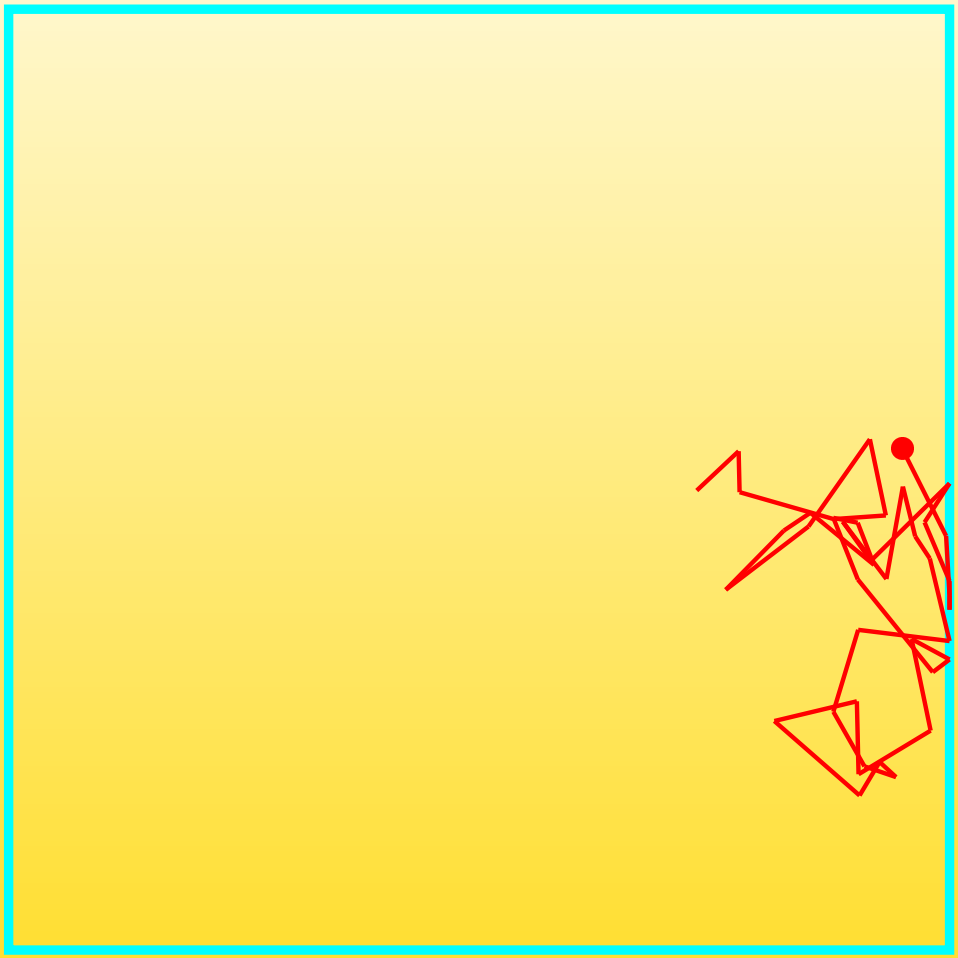


Figure 5: Random walk

End of animation

8 – Text shown through a lens

L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

Âme sentinelle
Murmurons l'aveu
De la nuit si nulle
Et du jour en feu.

Des humains suffrages
Des communs élans
Là tu te dégages
Et voles selon.

Puisque de vous seules,
Braises de satin,
Le devoir s'exhale
Sans qu'on dise : enfin.

Là pas d'espérance,
Nul orietur.
Science avec patience,
Le supplice est sûr.

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L'Éternité

Puisque de vous seules,
Braises de satin,
e devoir s'exhale
ans qu'on dise : enfin.

Là pas d'espérance,
Nul orietur.
Science avec patience,
Le supplice est sûr.

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

de me
De la nuit si nulle
Et du jour en feu.

Des humains suffrages
Des communs élans
Là tu te dégages
Et voles selon.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L’Éternité

Puisque de vous seules,
Braises de satin,
devoir s’exhale
s qu’on dise : enfin.

pas d’espérance,
ul orietur.
Science avec patience,
Le supplice est sûr.

Elle est retrouvée.
Quoi ? — L’Éternité.
C’est la mer allée
Avec le soleil.

Des humains suffrages
Des communs élans
Là tu te dégages
Et voles selon.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L’Éternité

Elle est retrouvée.
Quoi ?
C
er allée
oleil.

Puisque de vous seules,
Braises de satin,
devoir s’exhale
qu’on dise : enfin.

d’espérance,
etur.
ce avec patience,
supplice est sûr.

Des humains suffrages
Des communs élans
Là tu te dégages
Et voles selon.

Elle est retrouvée.
Quoi ? — L’Éternité.
C’est la mer allée
Avec le soleil.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L’Éternité

Elle est retrouvée.
Quoi ? — L’Éternité.
C’est la m
Avec

Puisque de vous seules,
Braises de satin,
Le devoir s’exhale
qu’on dise : enfin.

espérance,
vec patience,
ce est sûr.

es humains
Des communs élans
Là tu te dégages
Et voles selon.

Elle est retrouvée.
Quoi ? — L’Éternité.
C’est la mer allée
Avec le soleil.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L’Éternité

Elle est retrouvée.
Quoi ? — L’Éternité.
C’est la mer allée
Avec le soleil.

Âme
Mur
De
Et d

Des
D
a tu te dégages
Et voles selon.

Puisque de vous seules,
Braises de satin,
Le devoir s’exhale
Sans qu’on dise : enfin.

spérance,

patience,
est sûr.

ou

etrouvée.
oi ? — L’Éternité.
C’est la mer allée
Avec le soleil.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

Âme sentin[illegible]spérance,
Murmur[illegible]
De la n[illegible]atience,
Et du [illegible] sûr.

Des hun[illegible]vée.
Des com[illegible]Éternité.
Là tu[illegible]égage[illegible] mer allée
E[illegible]es selon.

Puisque de vous seules,
Braises de satin,
Le devoir s'exhale
Sans qu'on dise : enfin.

Nu

S

Avec le soleil.

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

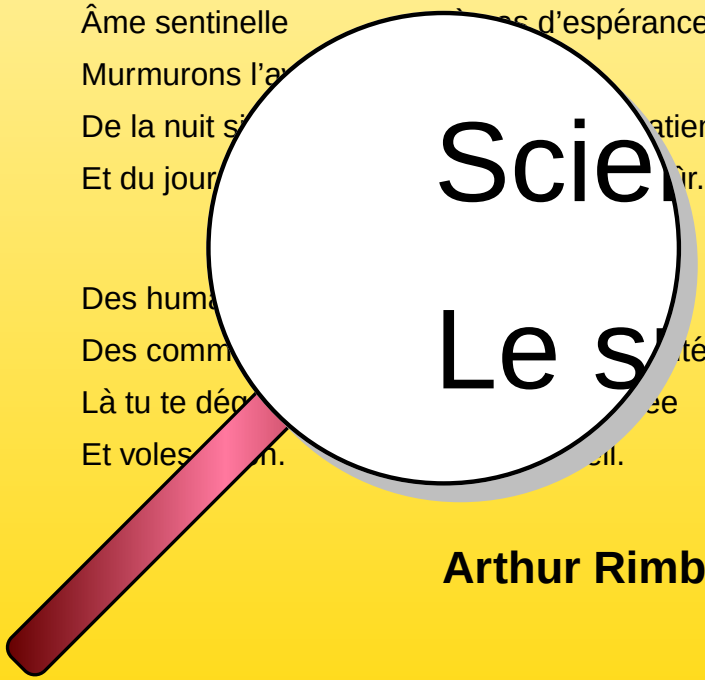
Puisque de vous seules,
Braises de satin,
Le devoir s'exhale
Sans qu'on dise : enfin.

Âme sentinelle
Murmurons l'a
De la nuit si
Et du jour

d'espérance,
atience,
r.

Des huma
Des comm
Là tu te déc
Et voles

té.
ee
n.
il.



Arthur Rimbaud

End of animation

8 – Text shown through a lens

L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

Âme sentinelle
Murmurons l'aveu
De la nuit si nulle
Et du jour en fe

Des humain
Des commu
Là tu te déga
Et voles selon.

Puisque de vous seules,
Braises de satin,
Le devoir s'exhale
Sans qu'on dise : enfin.

Là pas d'espérance,
Nul orietur.
patience,
r.

Le sup

Arthur Rimbaud

End of animation

8 – Text shown through a lens

L'Éternité

Elle est retrouvée.
Quoi ? — L'Éternité.
C'est la mer allée
Avec le soleil.

Âme sentinelle
Murmurons l'aveu
De la nuit si nulle
Et du jour en feu.

Des humains su
Des communs
Là tu te dégag
Et voles selon.

Puisque de vous seules,
Braises de satin,
Le devoir s'exhale
Sans qu'on dise : enfin.

Là pas d'espérance,
Nul orietur.
Science avec patience,
sûr.

lle est re

baud

End of animation

8 – Text shown through a lens

L’Éternité

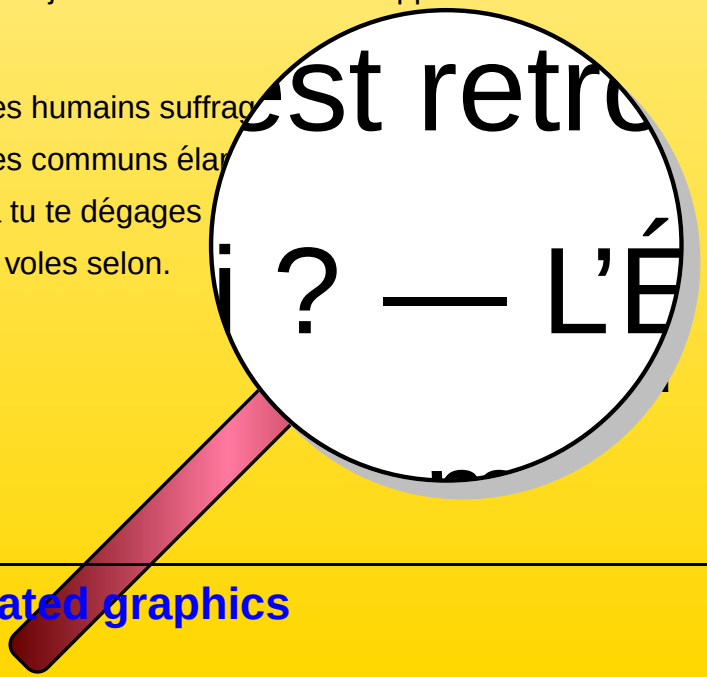
Elle est retrouvée.
Quoi ? — L’Éternité.
C’est la mer allée
Avec le soleil.

Âme sentinelle
Murmurons l’aveu
De la nuit si nulle
Et du jour en feu.

Des humains suffrag
Des communs élan
Là tu te dégages
Et voles selon.

Puisque de vous seules,
Braises de satin,
Le devoir s’exhale
Sans qu’on dise : enfin.

Là pas d’espérance,
Nul orietur.
Science avec patience,
Le supplice est sûr.



End of animation

9 – Text progressively shown

End of animation

9 – Text progressively shown

D

End of animation

9 – Text progressively shown

Do

End of animation

9 – Text progressively shown

Do

End of animation

9 – Text progressively shown

Do n

End of animation

9 – Text progressively shown

Do no

End of animation

9 – Text progressively shown

Do not

End of animation

9 – Text progressively shown

Do not

End of animation

9 – Text progressively shown

Do not d

End of animation

9 – Text progressively shown

Do not do

End of animation

9 – Text progressively shown

Do not do

End of animation

9 – Text progressively shown

Do not do I

End of animation

9 – Text progressively shown

Do not do li

End of animation

9 – Text progressively shown

Do not do lik

End of animation

9 – Text progressively shown

Do not do like

End of animation

9 – Text progressively shown

Do not do like

End of animation

9 – Text progressively shown

Do not do like m

End of animation

9 – Text progressively shown

Do not do like me

End of animation

9 – Text progressively shown

Do not do like me!

End of animation

10 – Text progressively vanished

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

10 – Text progressively vanished

Oh! my dear friends...

It is time to tell you

good bye!

See you again soon!

End of animation

11 – Building of a regular polygon of seventeen sides

Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

1: Definition of the center O of the polygon

Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

2: Definition of the point P_1 at 5 units from O

Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

3: Circle of center O with the point P_1 on it

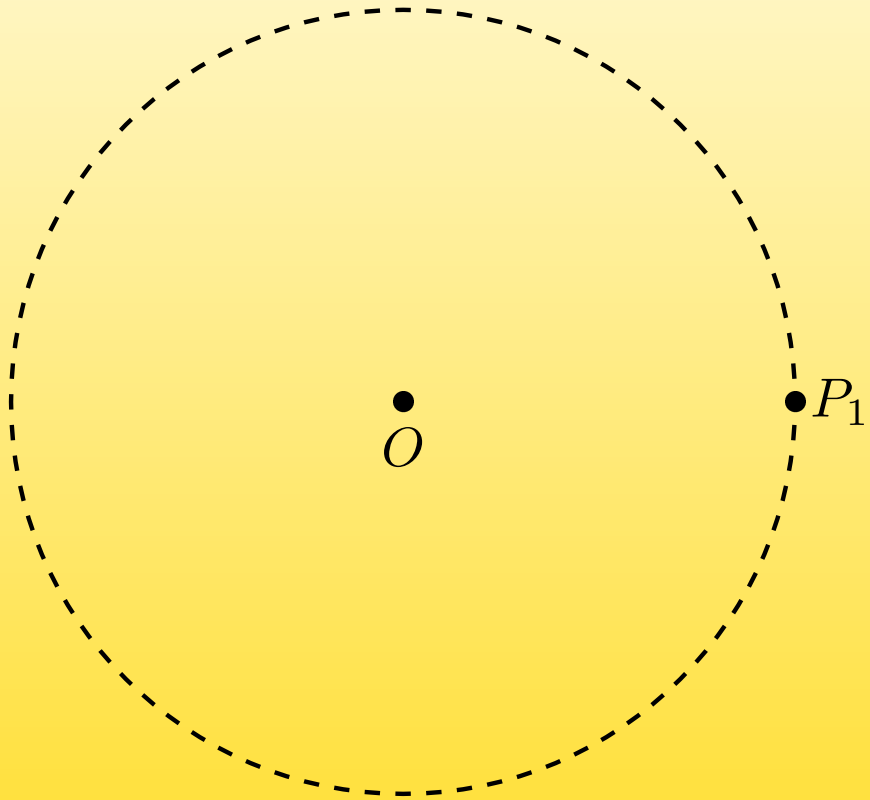


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

4: Definition of the point PP_1, symmetric to the point P_1

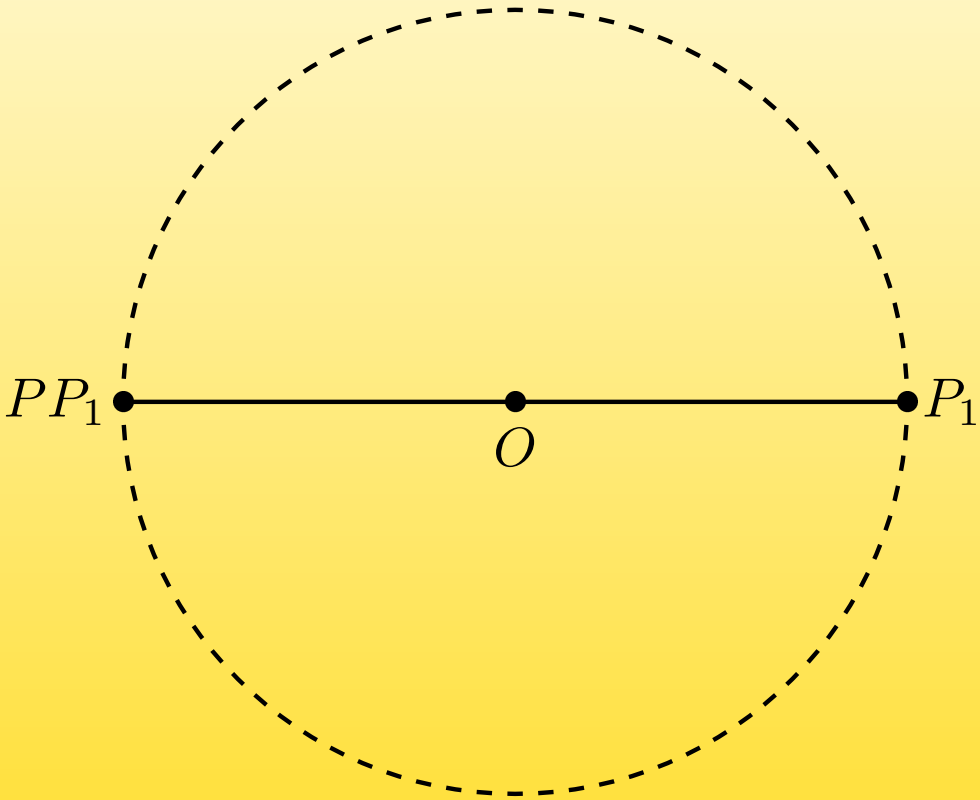


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

5: Definition of the point B, with P_1-O-B a right angle

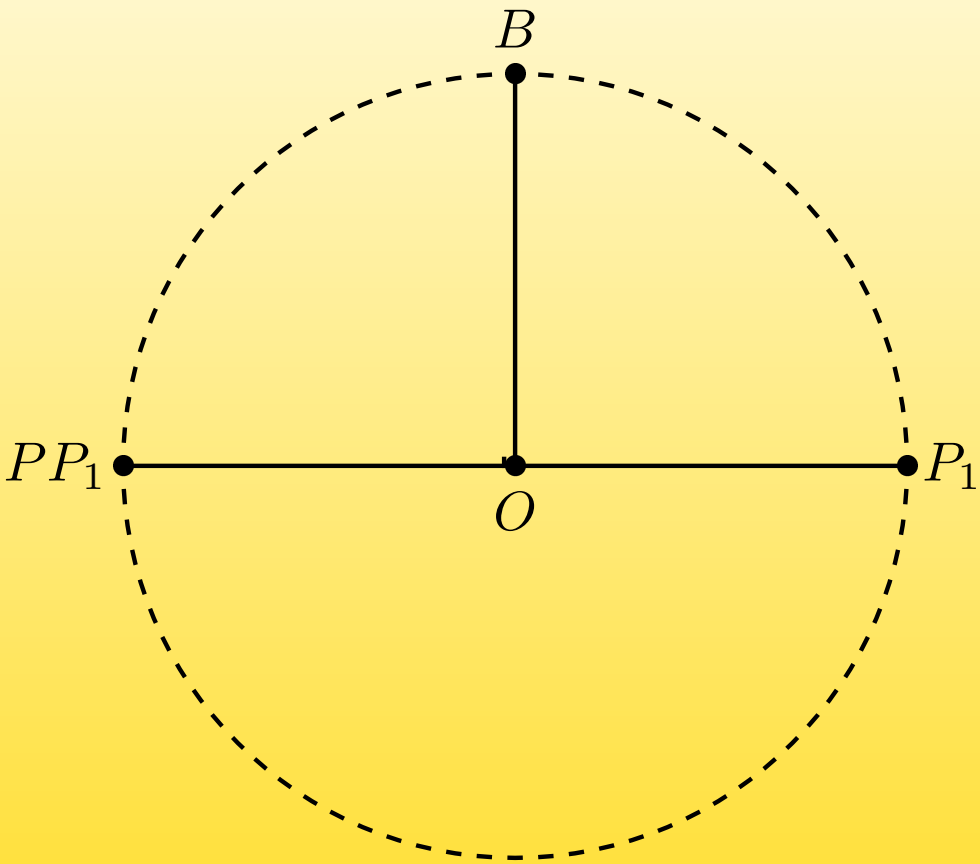


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

6: Definition of the point J, as 0.25 of O-B

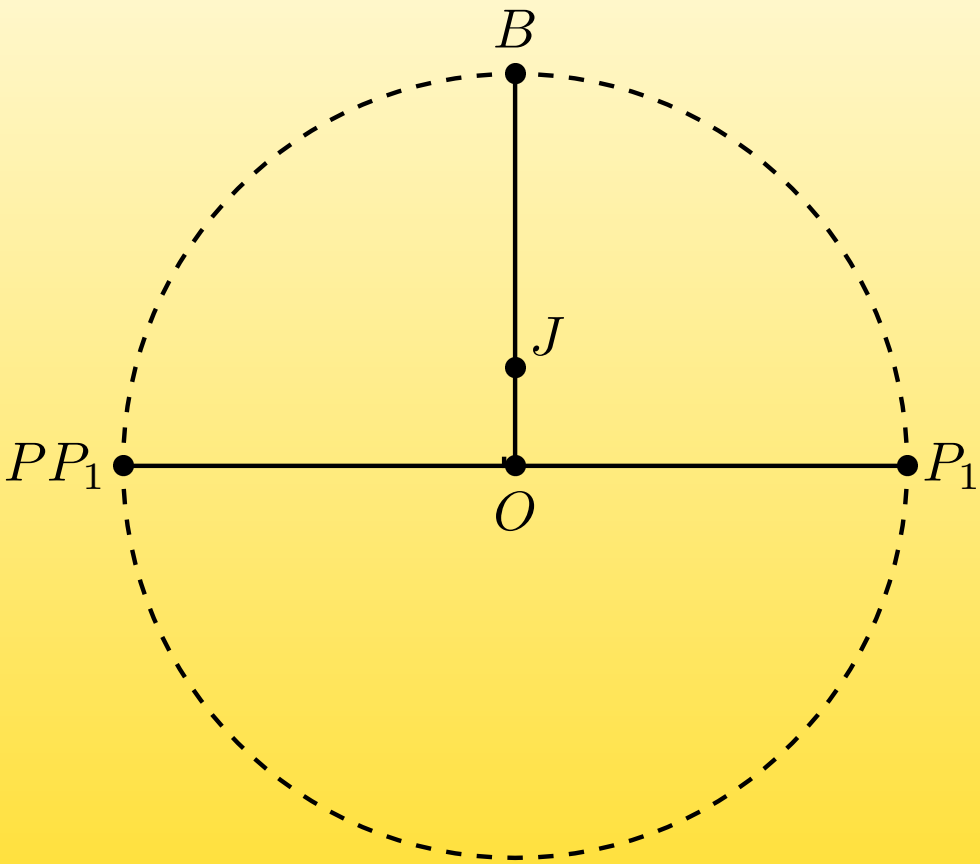


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

7: Line between the points J and P_1

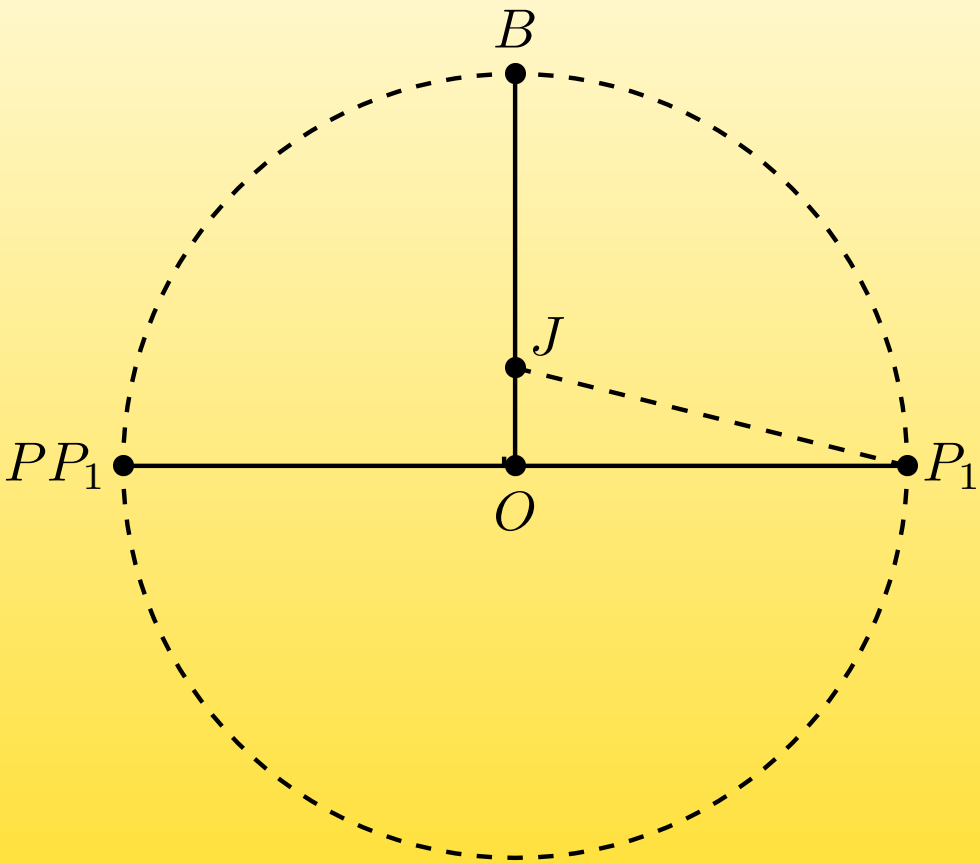


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

8: Bissectrisse of the angle defined by the points J, O, and P_1, which define the point PE1

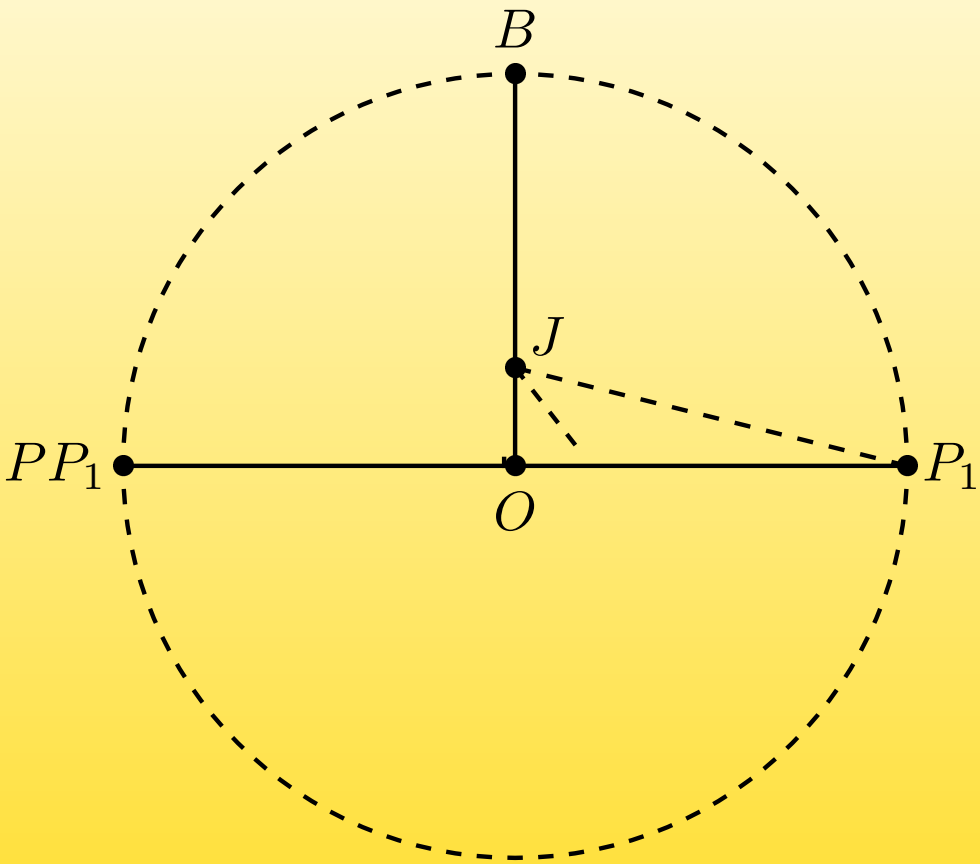


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

9: Bissectrisse of the angle defined by the points J, O, and PE1, which define the point PE2

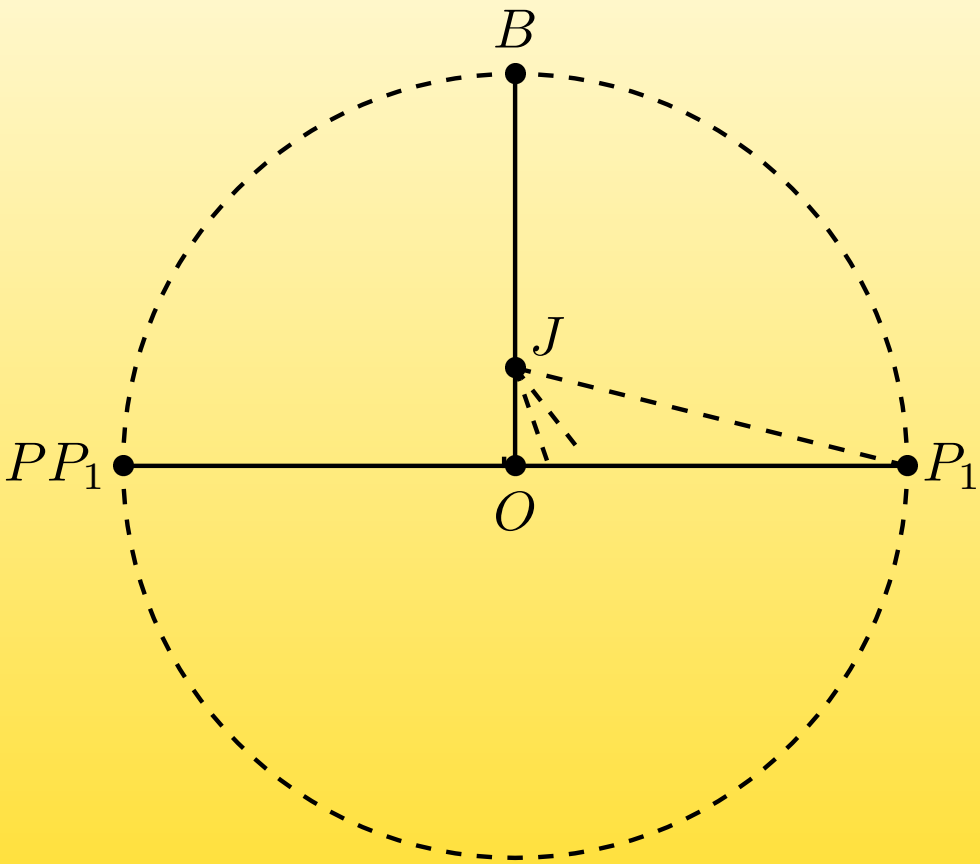


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

10: Definition of the point E, as intersection of the two lines O-P_1 and J-PE2

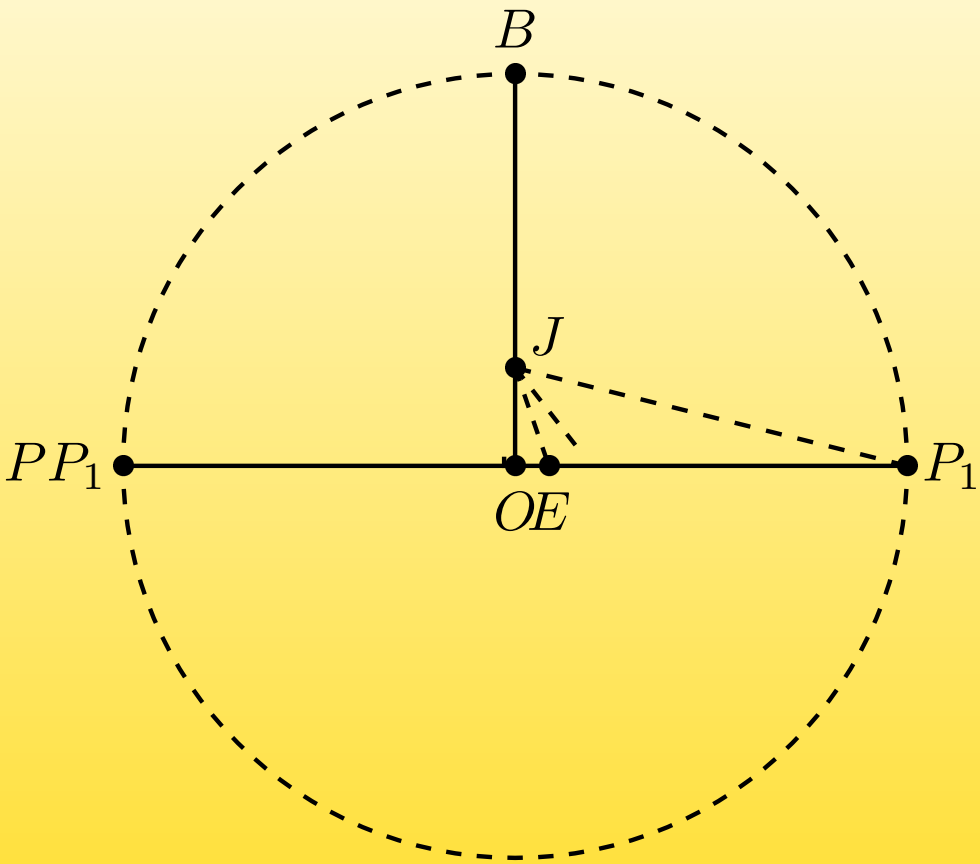


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

11: Definition of the point F, as intersection of the two lines O-P_1 and J-PF1, with PF1 defined by J and E

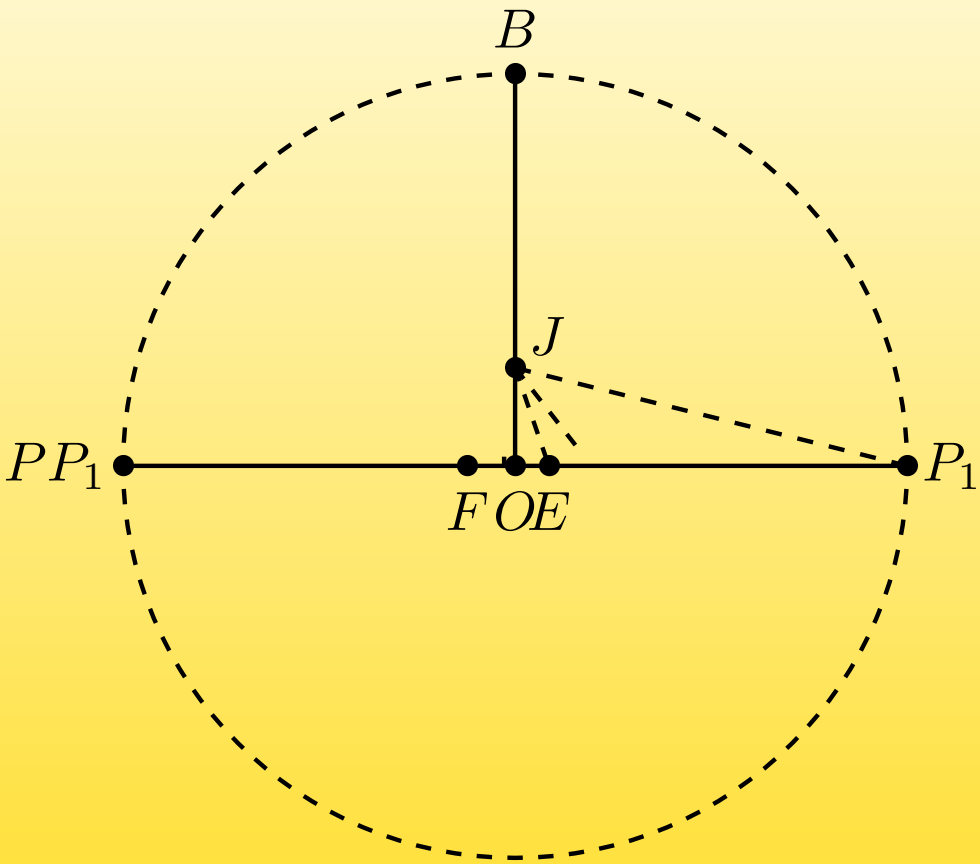


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

12: Definition of the point MFP1, as middle of the line F-P_1

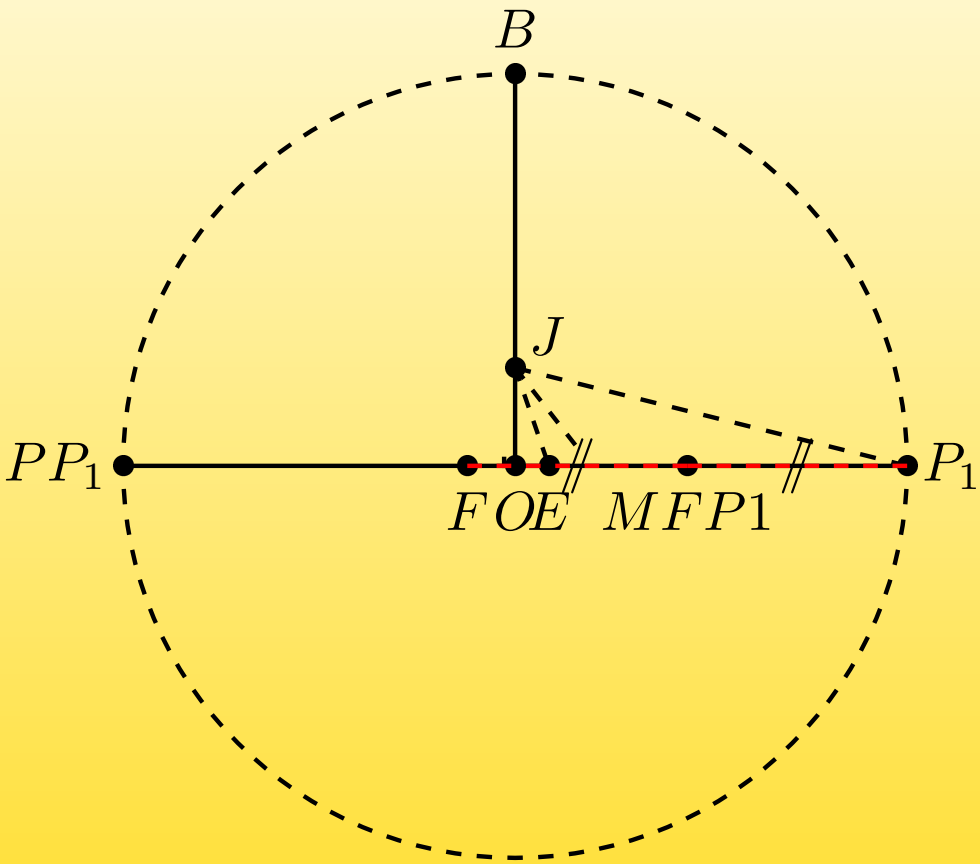


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

13: Circle of center MFP1 with point P_1 on it

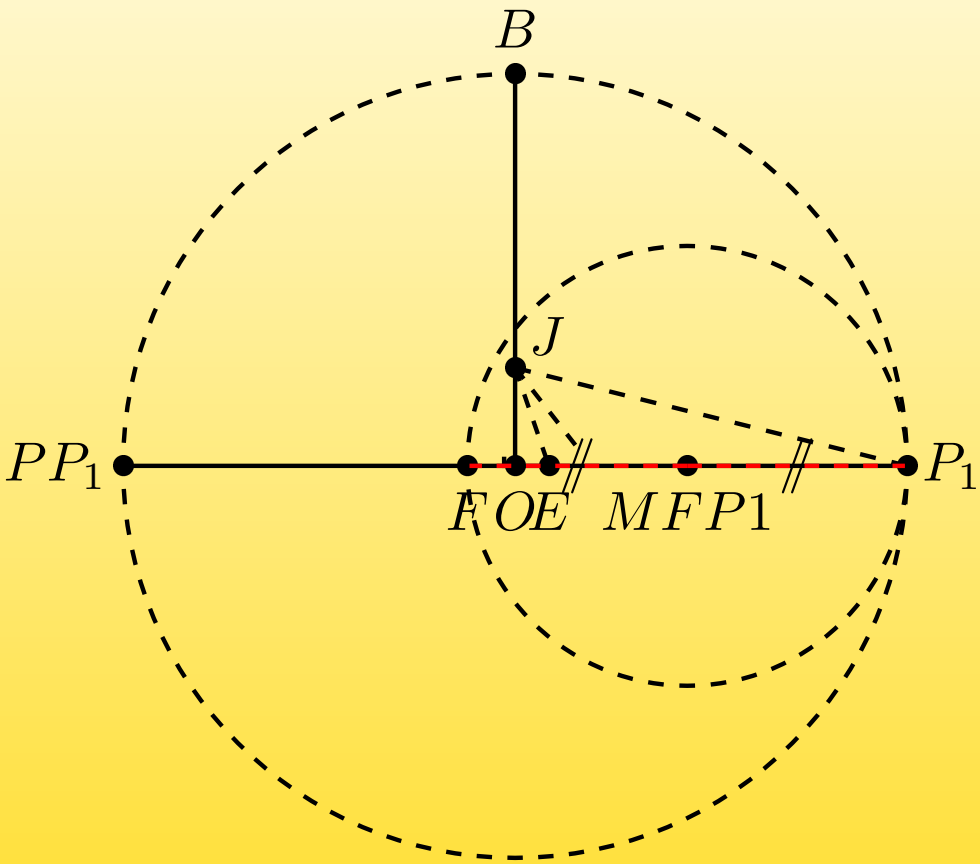


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

14: Definition of the point K, as intersection of the line O-B and the circle of center MFP1 and radius P_1

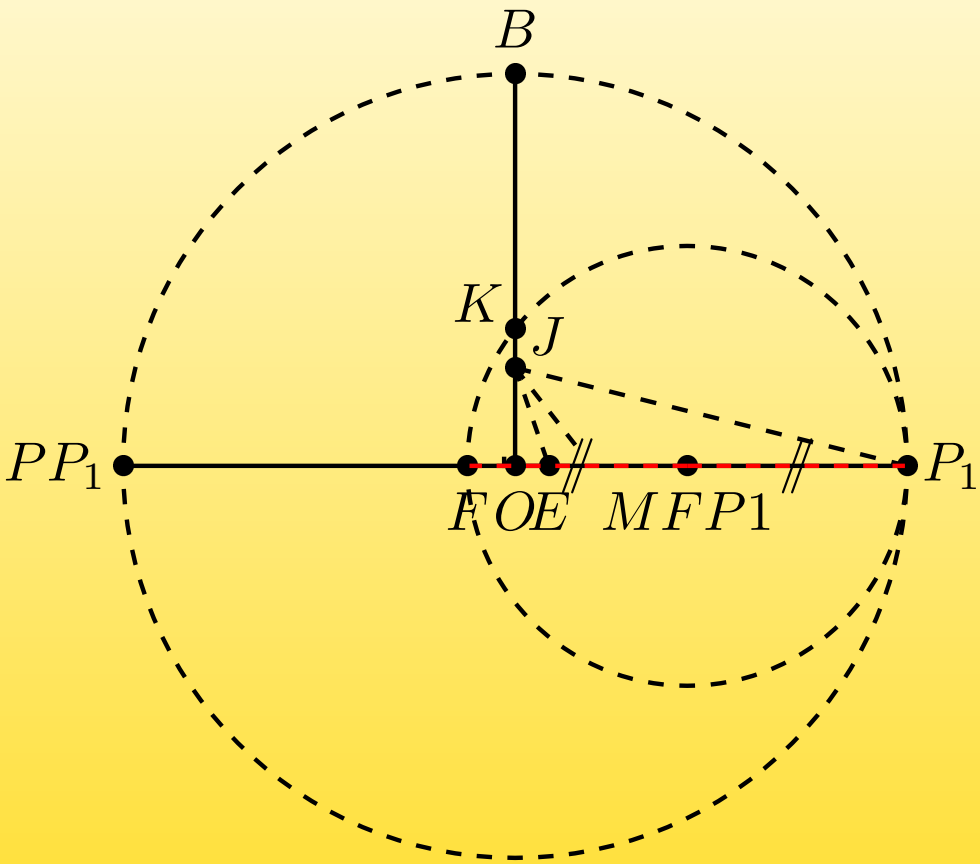


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

15: Circle of center E with point K on it

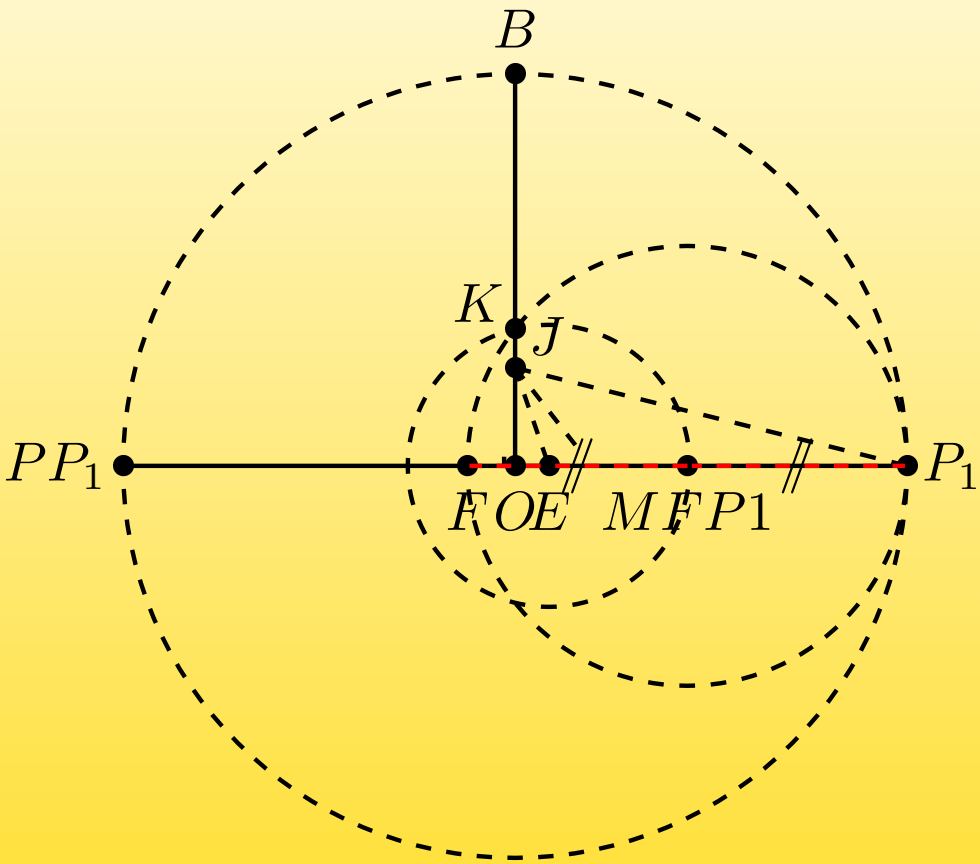


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

16: Definition of the points N_4 and N_6, as intersection of the line P_1-E and the circle of center E radius K

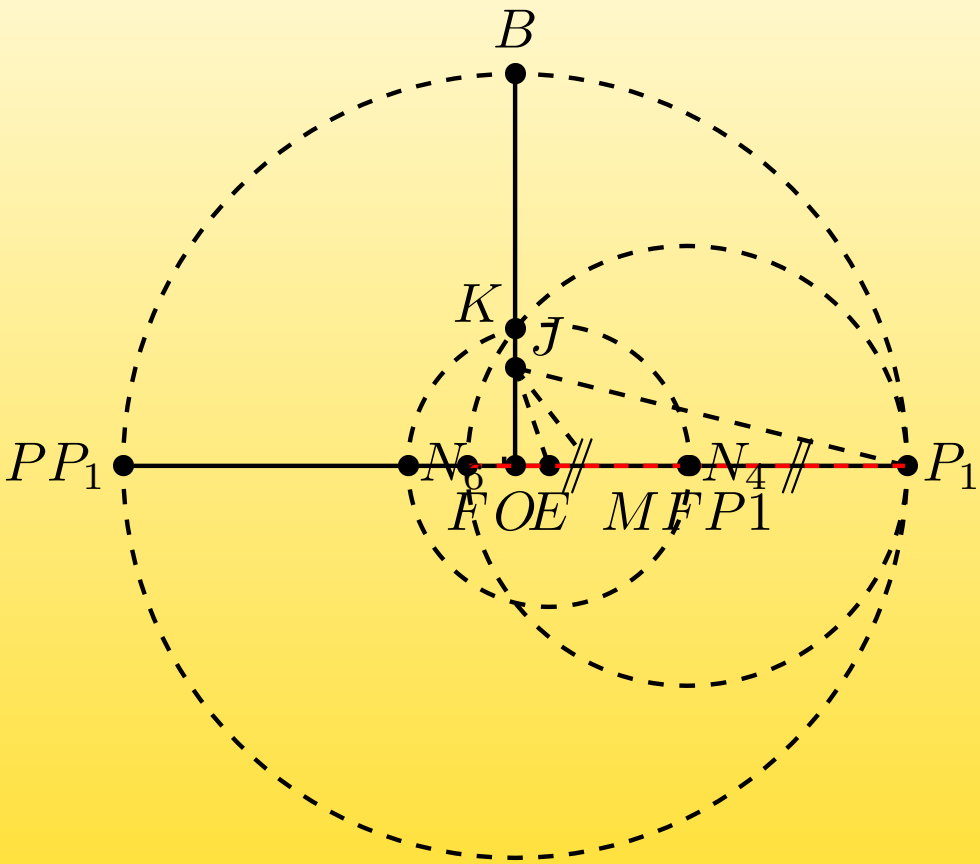


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

17: Definition of the points P_6 and P_13, as intersection of the line N_6-PP_6 and the original circle

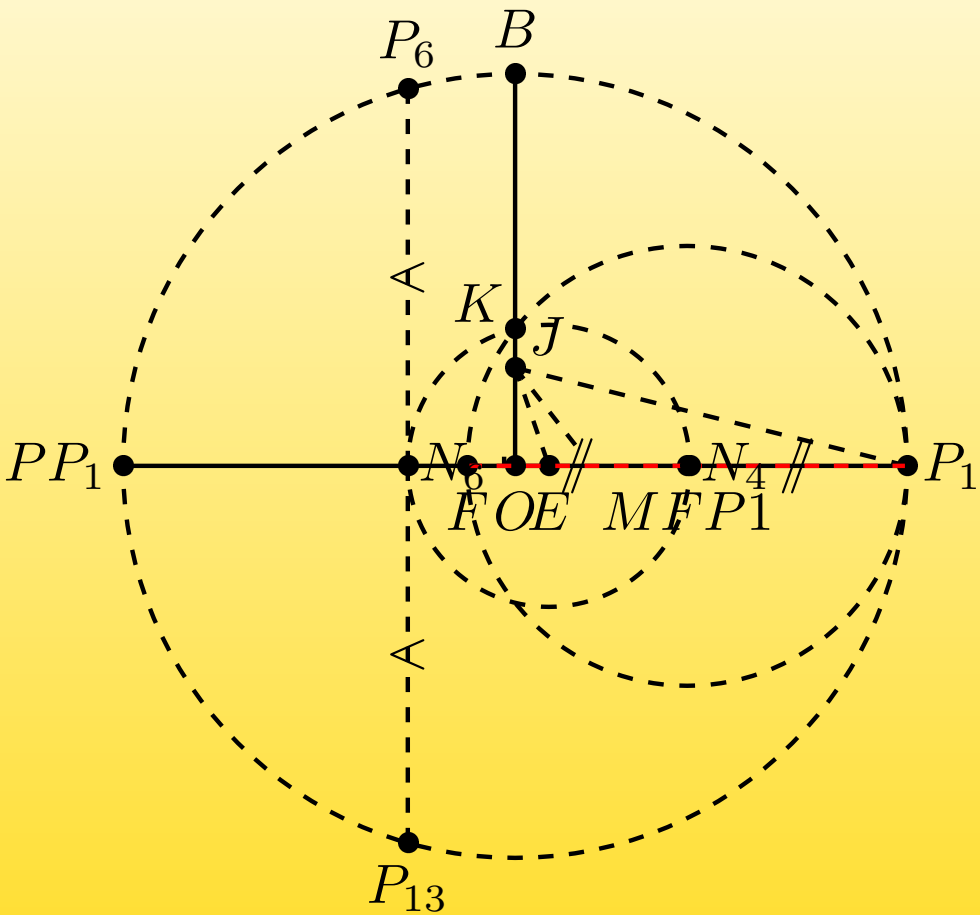


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

18: Definition of the points P_4 and P_15, as intersection of the line N_4-PP_4 and the original circle

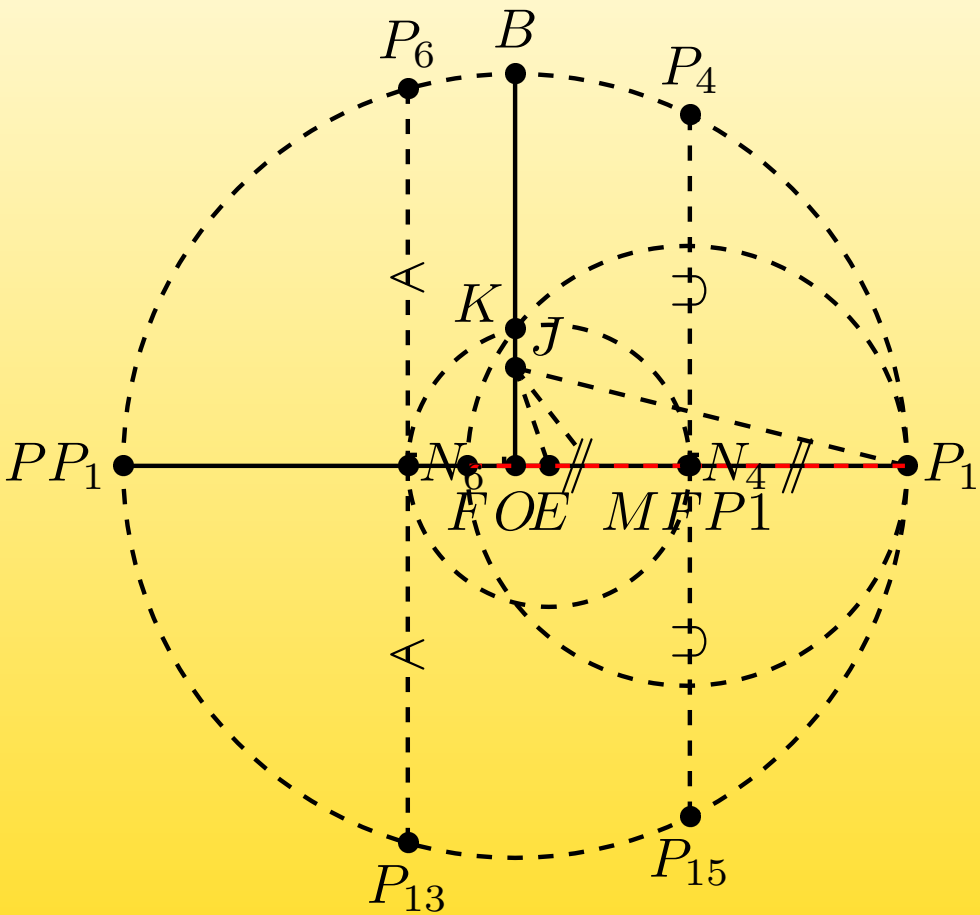


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

19: Bissectrisse of the angle defined by the points P_4, O, and P_6, which define the point P_5

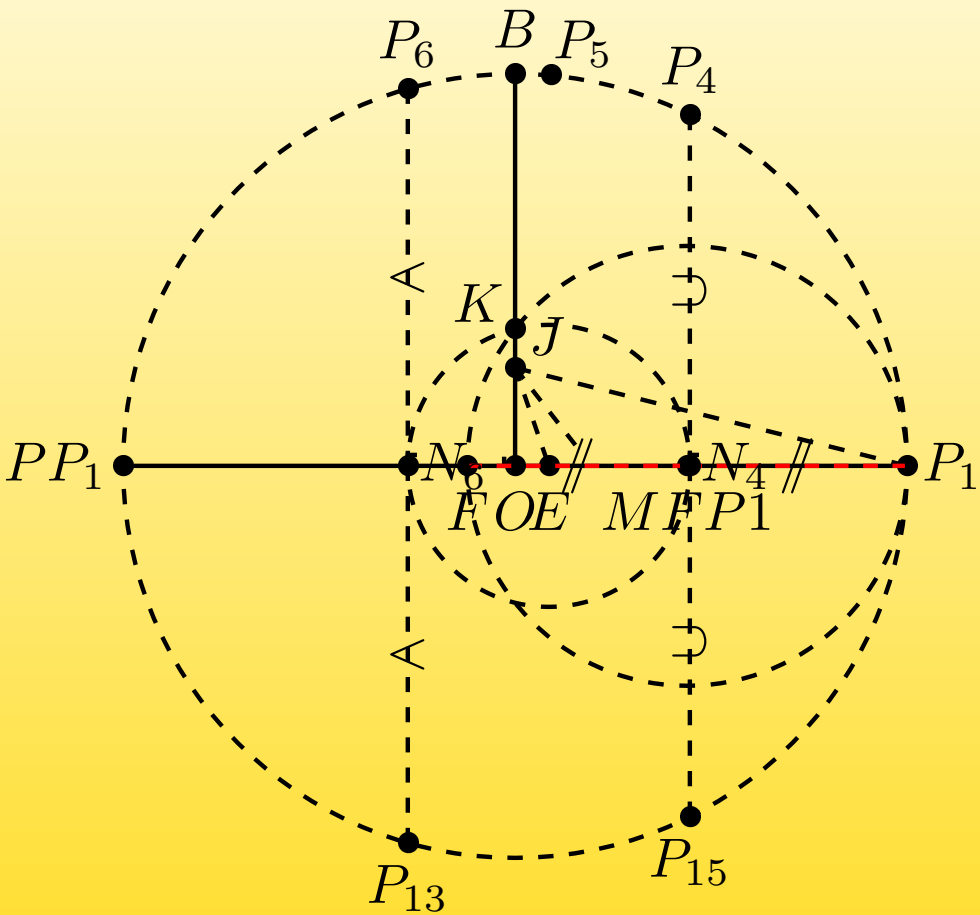


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

20: Definition of the point P_14 on the original circle, by orthogonal symmetry with the point P_5

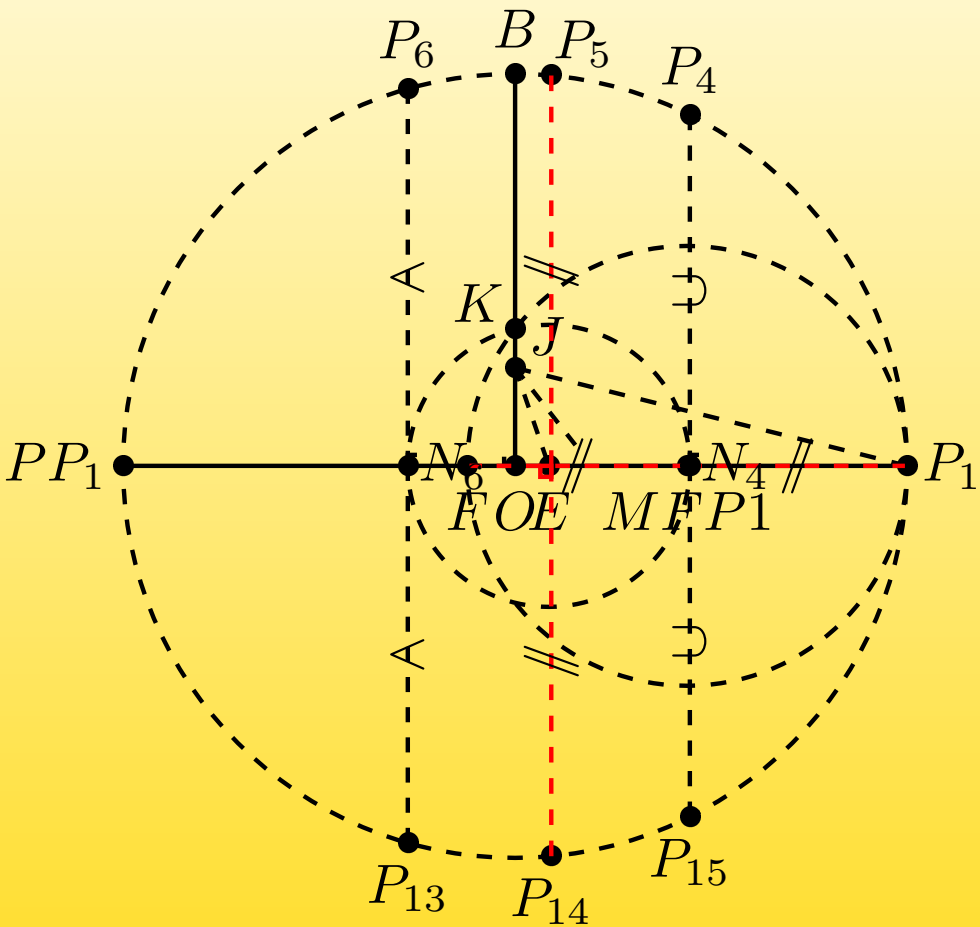


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

21: Definition of the point P_3 on the original circle, by intersection of two circles

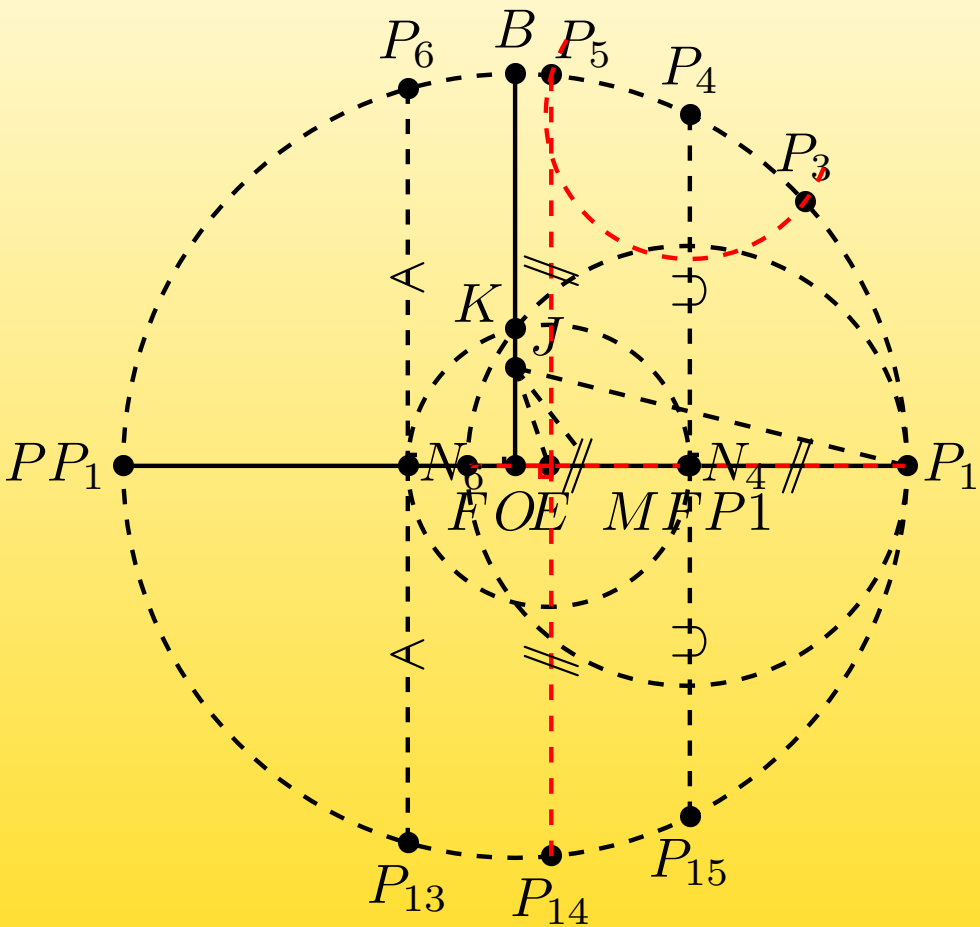


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

22: Definition of the point P_16 on the original circle, by orthogonal symmetry with the point P_3

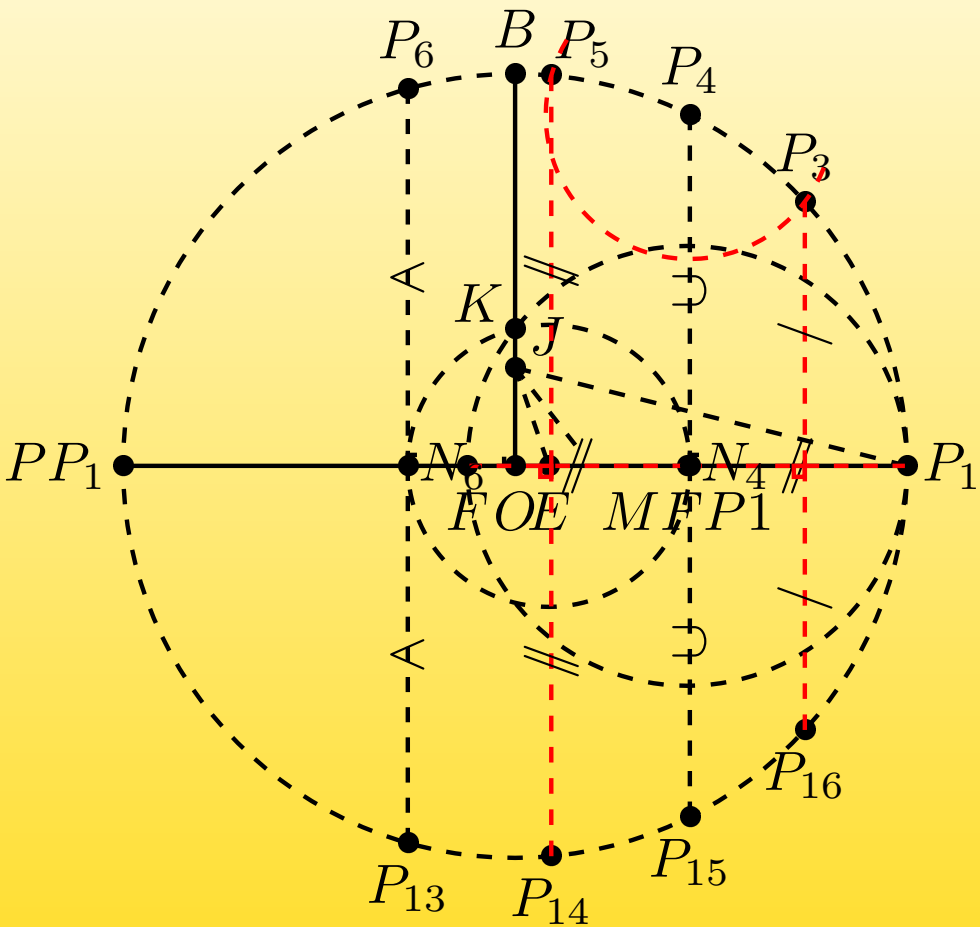


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

23: Definition of the point P_2 on the original circle, by intersection of two circles

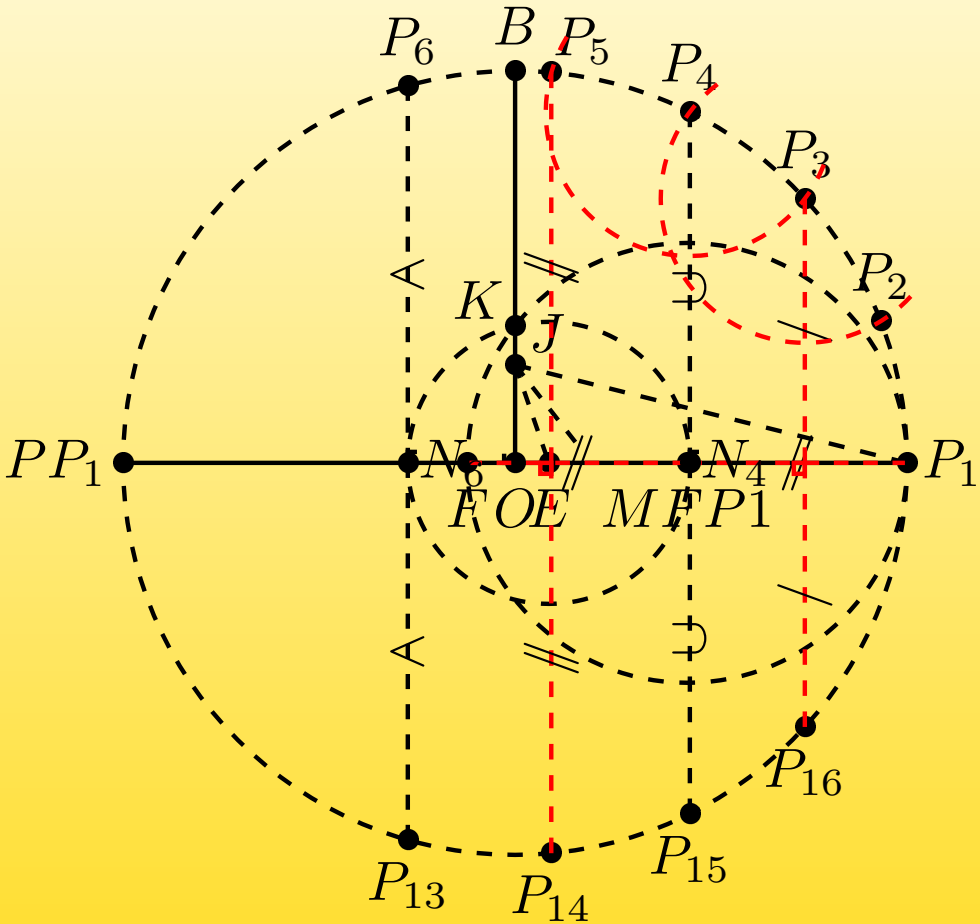


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

24: Definition of the point P_17 on the original circle, by orthogonal symmetry with the point P_2

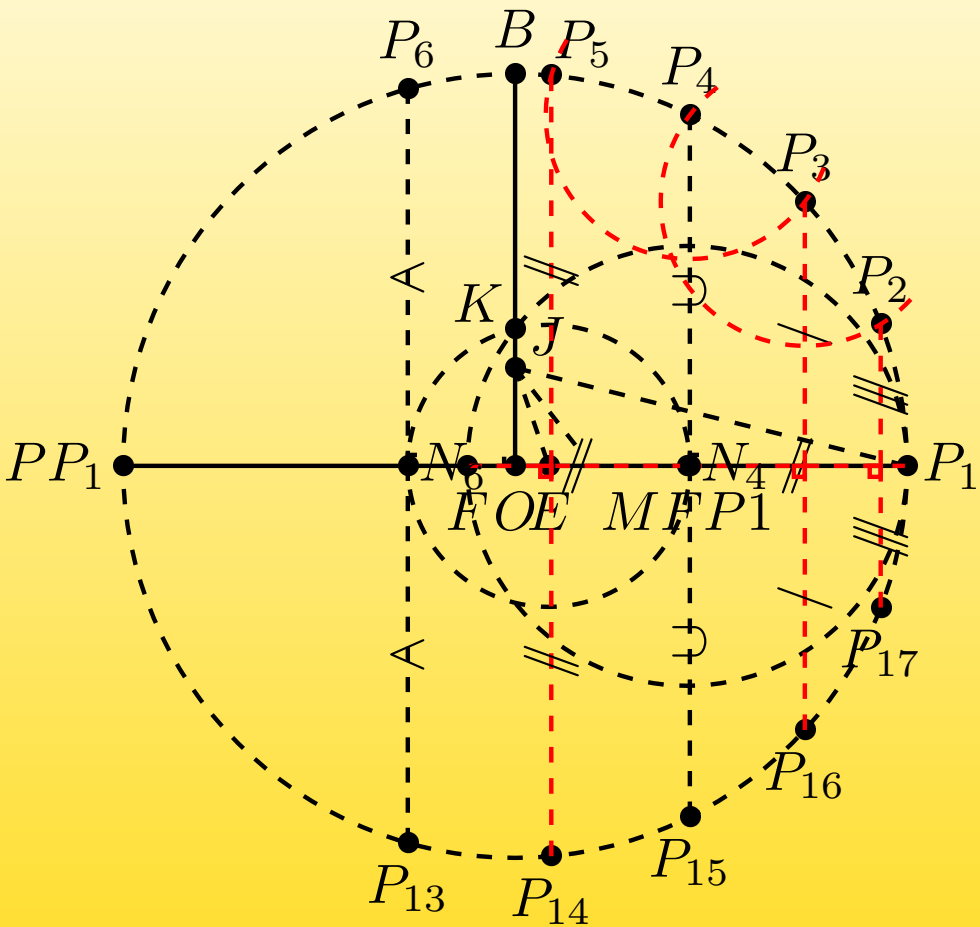


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

25: Definition of the point P_7 on the original circle, by intersection of two circles

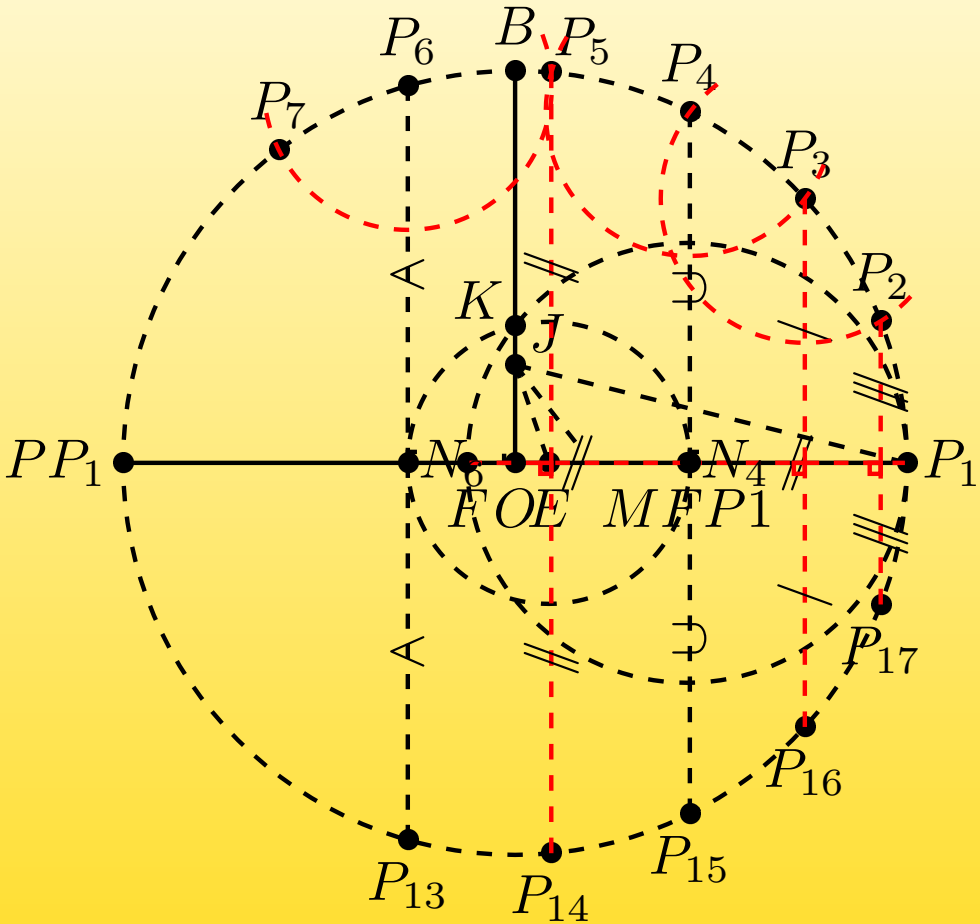


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

26: Definition of the point P_12 on the original circle, by orthogonal symmetry with the point P_7

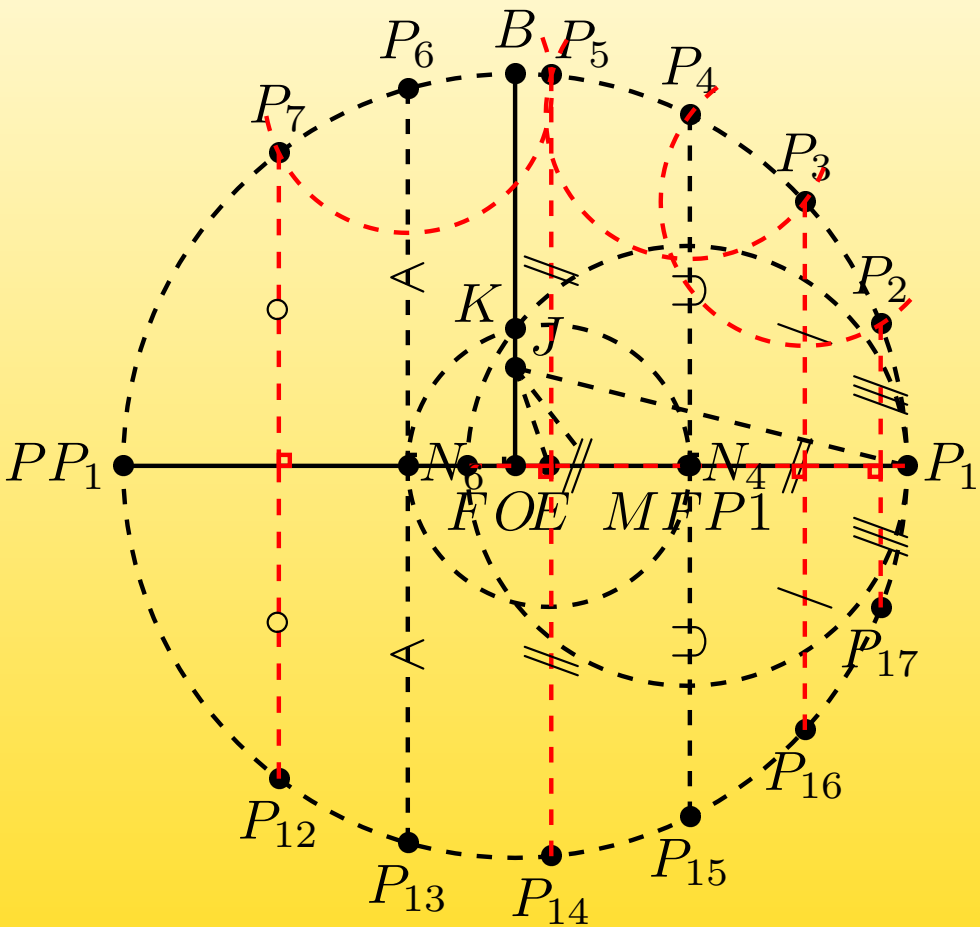


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

27: Definition of the point P_8 on the original circle, by intersection of two circles

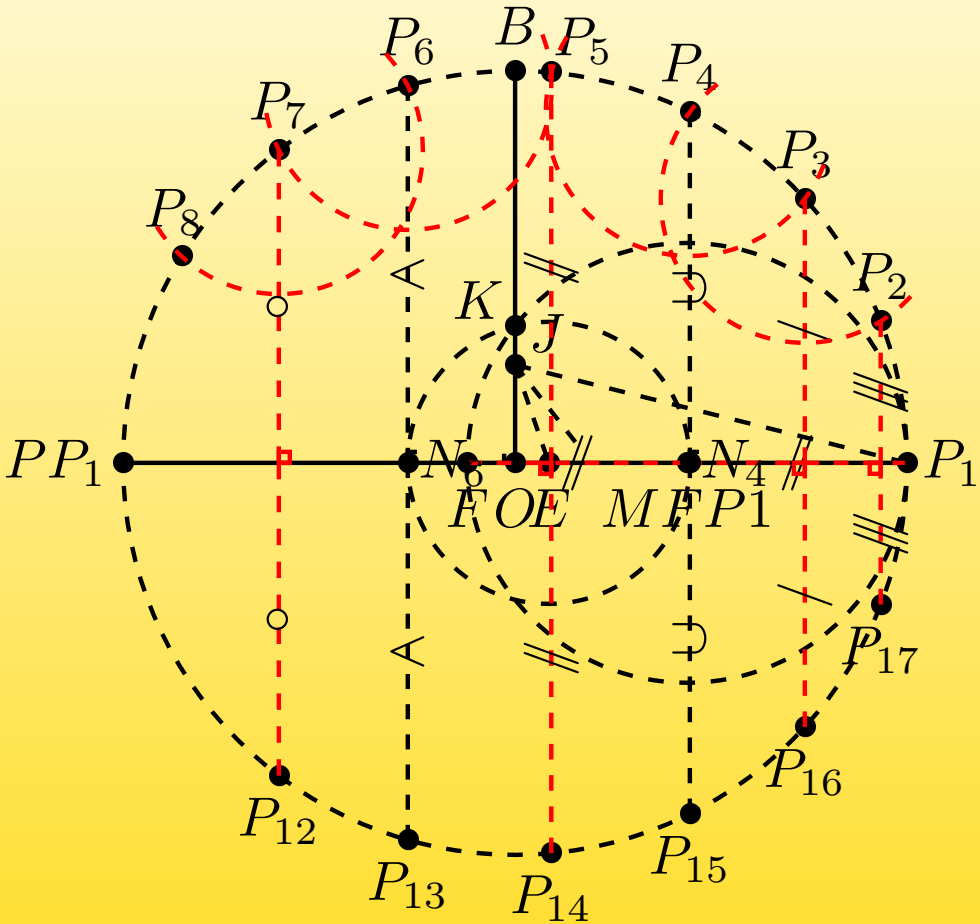


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

28: Definition of the point P_11 on the original circle, by orthogonal symmetry with the point P_8

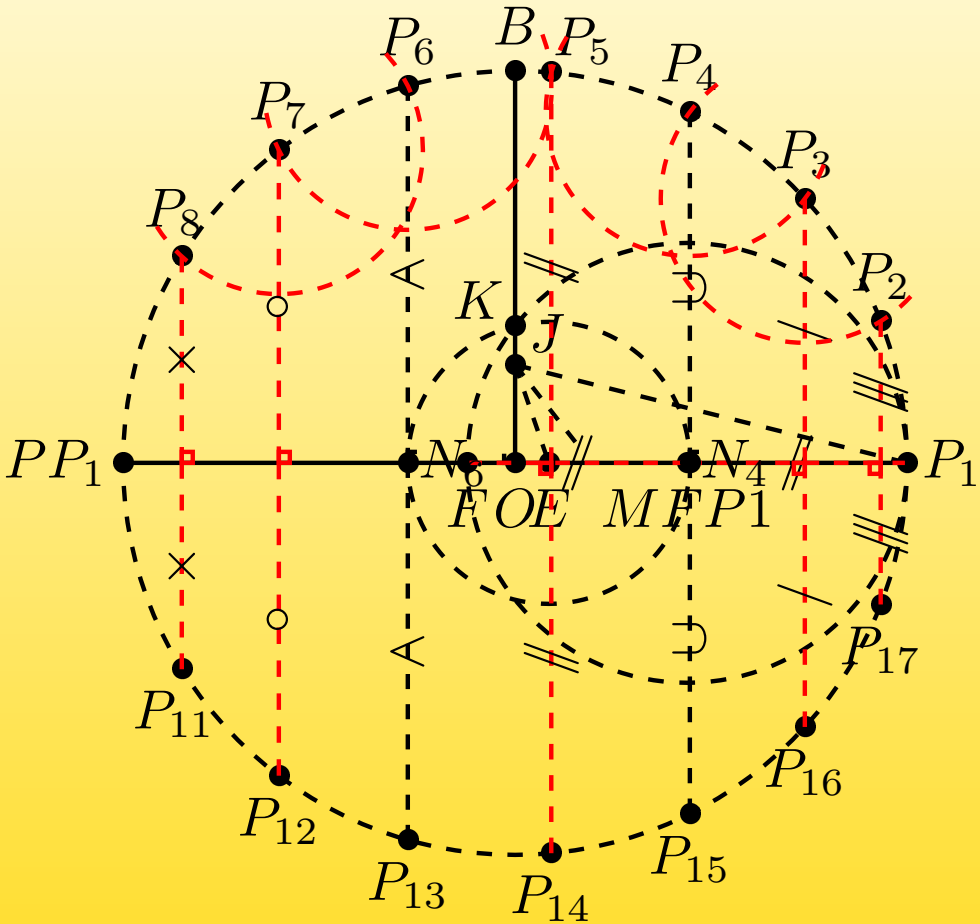


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

29: Definition of the point P_9 on the original circle, by intersection of two circles

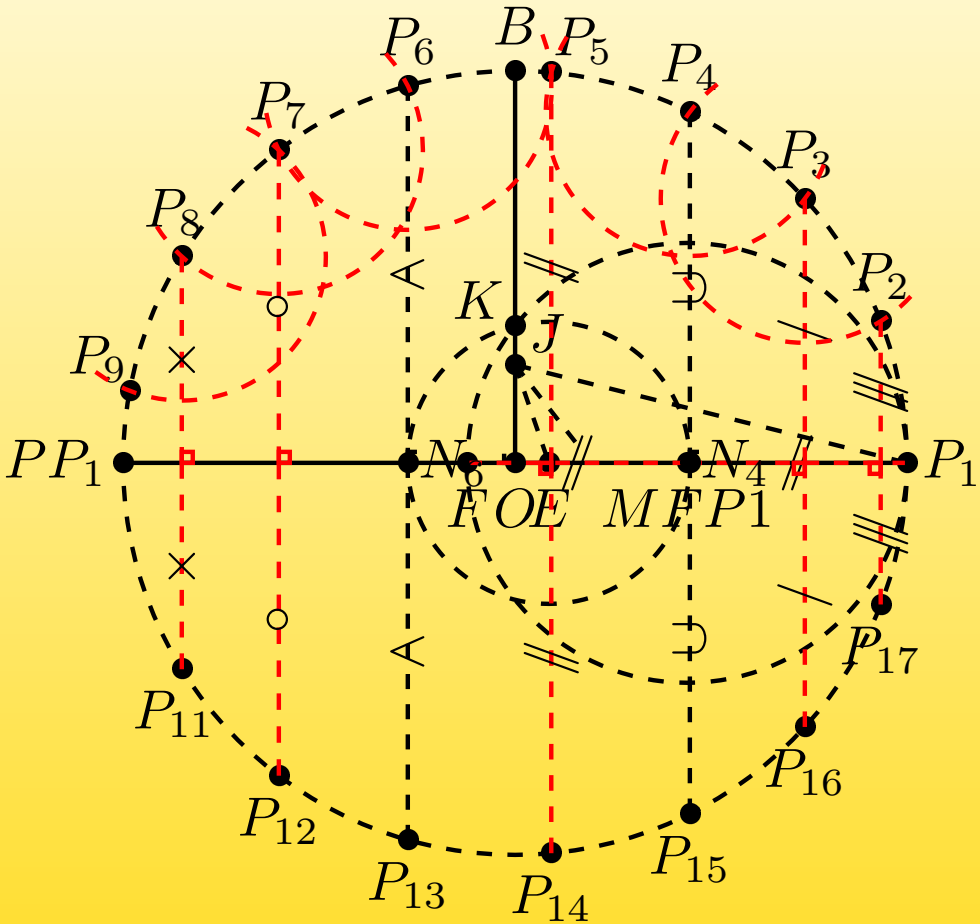


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

30: Definition of the point P_10 on the original circle, by orthogonal symmetry with the point P_9

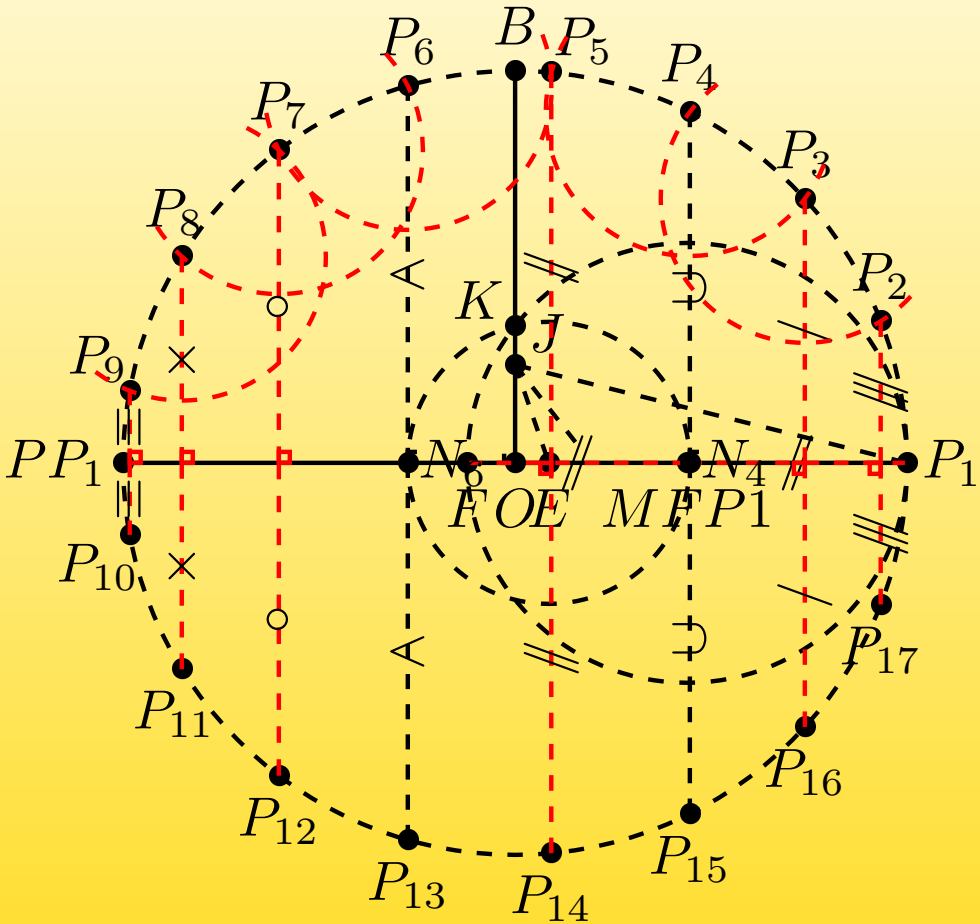


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

31: Side number 1 of the polygon

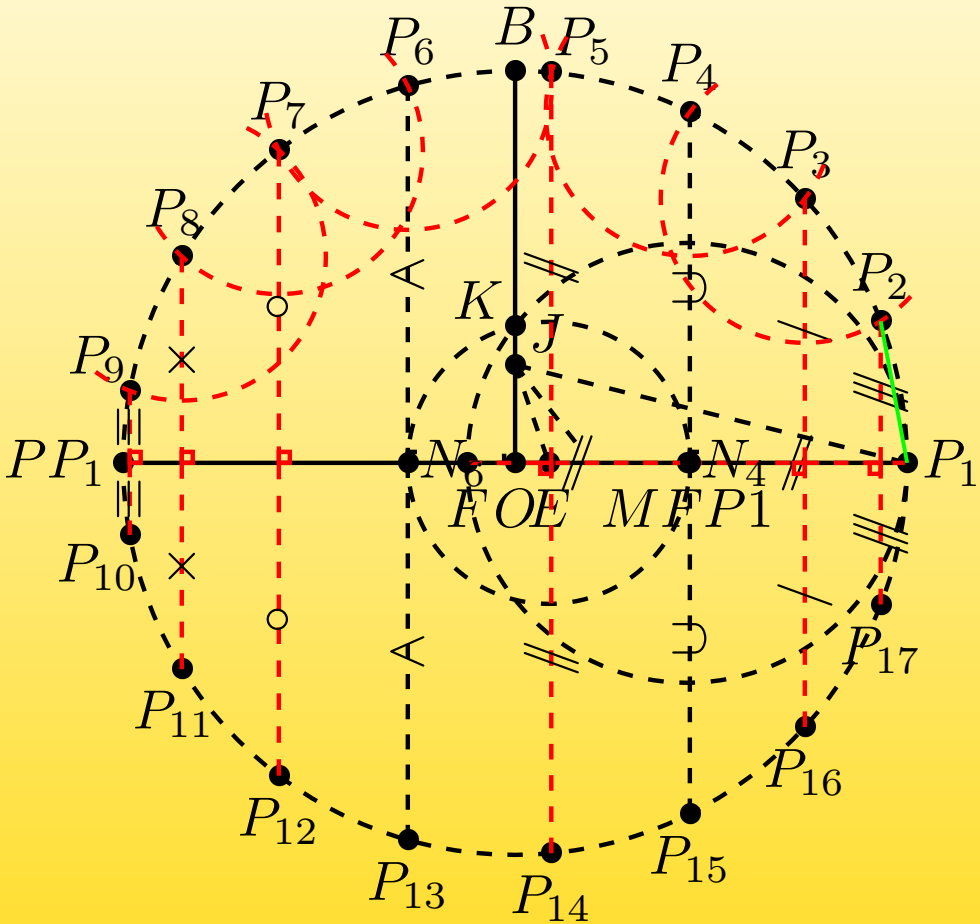


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

32: Side number 2 of the polygon

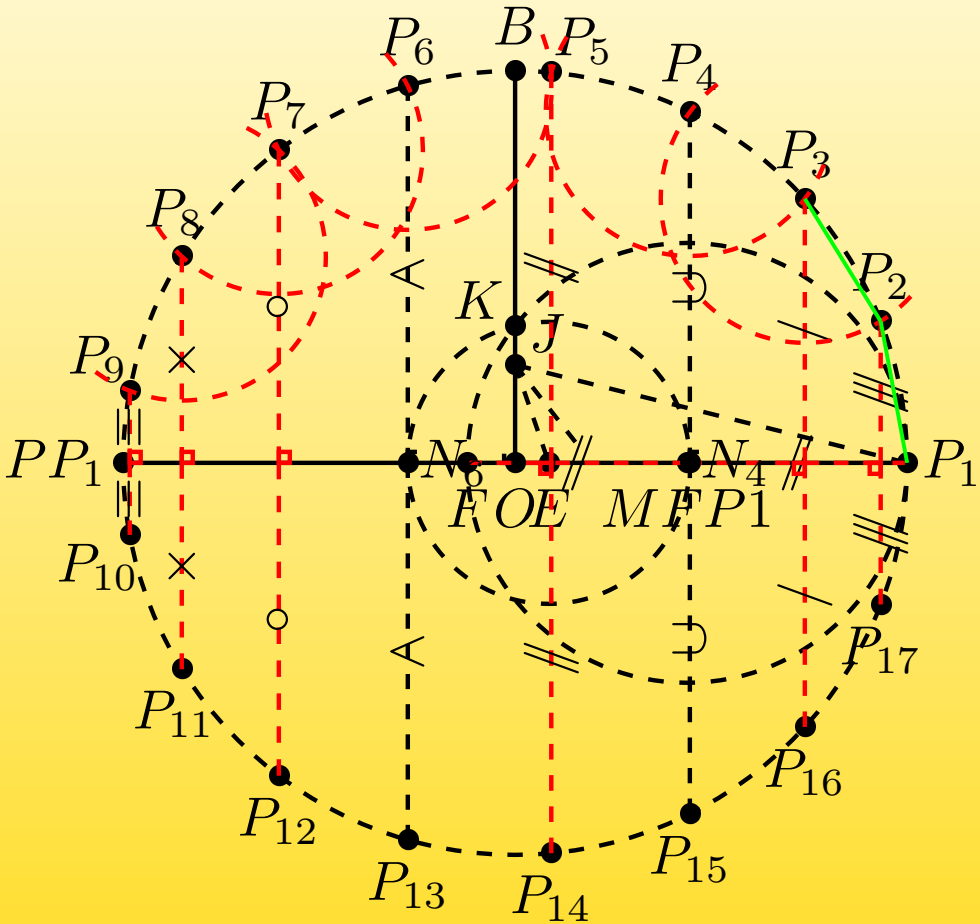


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

33: Side number 3 of the polygon

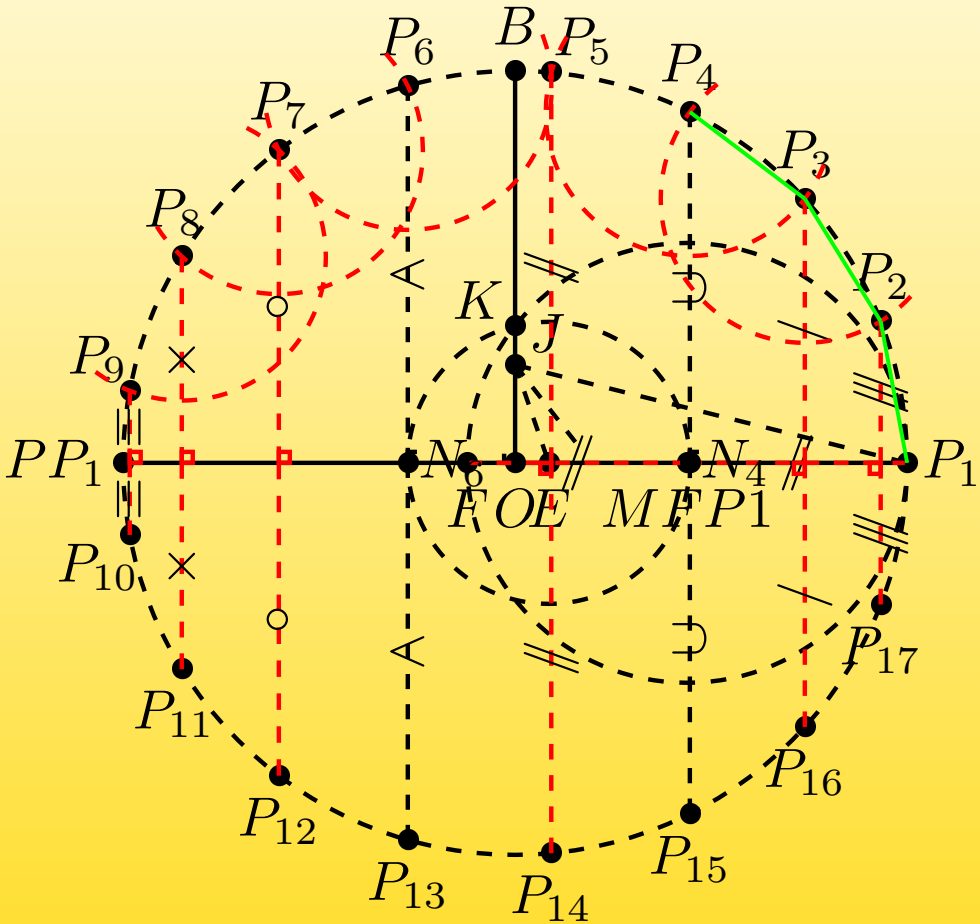


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

34: Side number 4 of the polygon

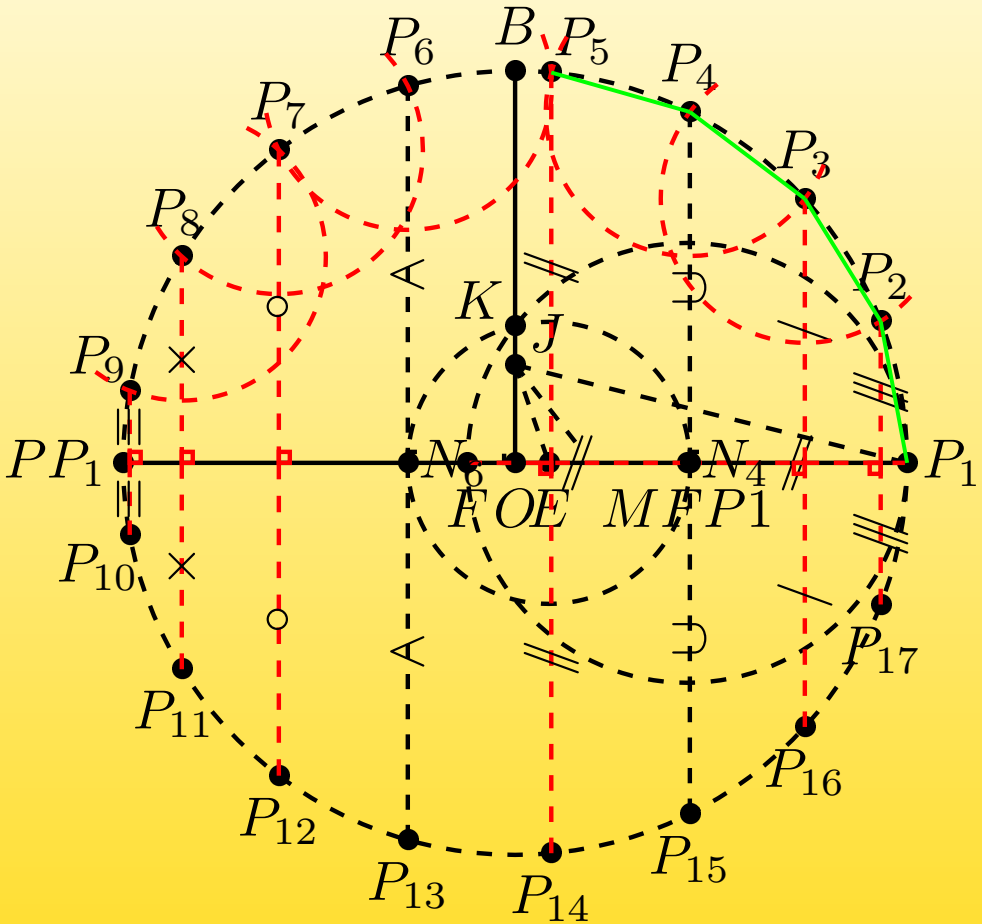


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

35: Side number 5 of the polygon

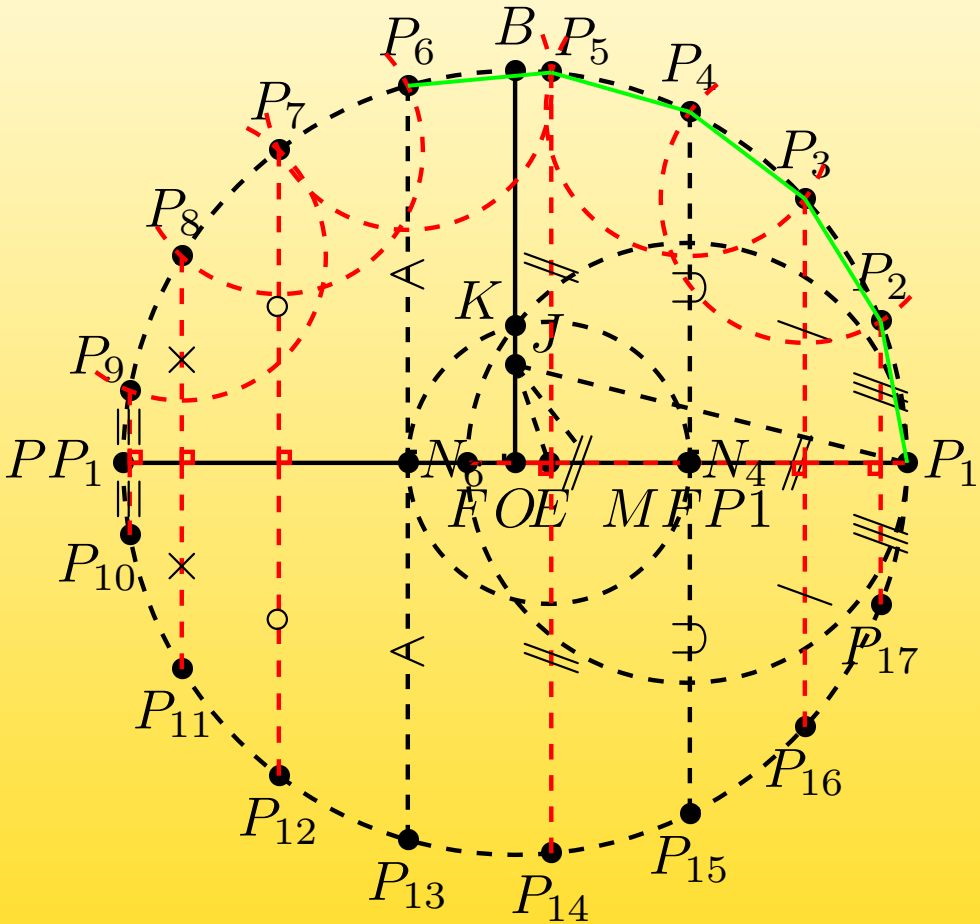


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

36: Side number 6 of the polygon

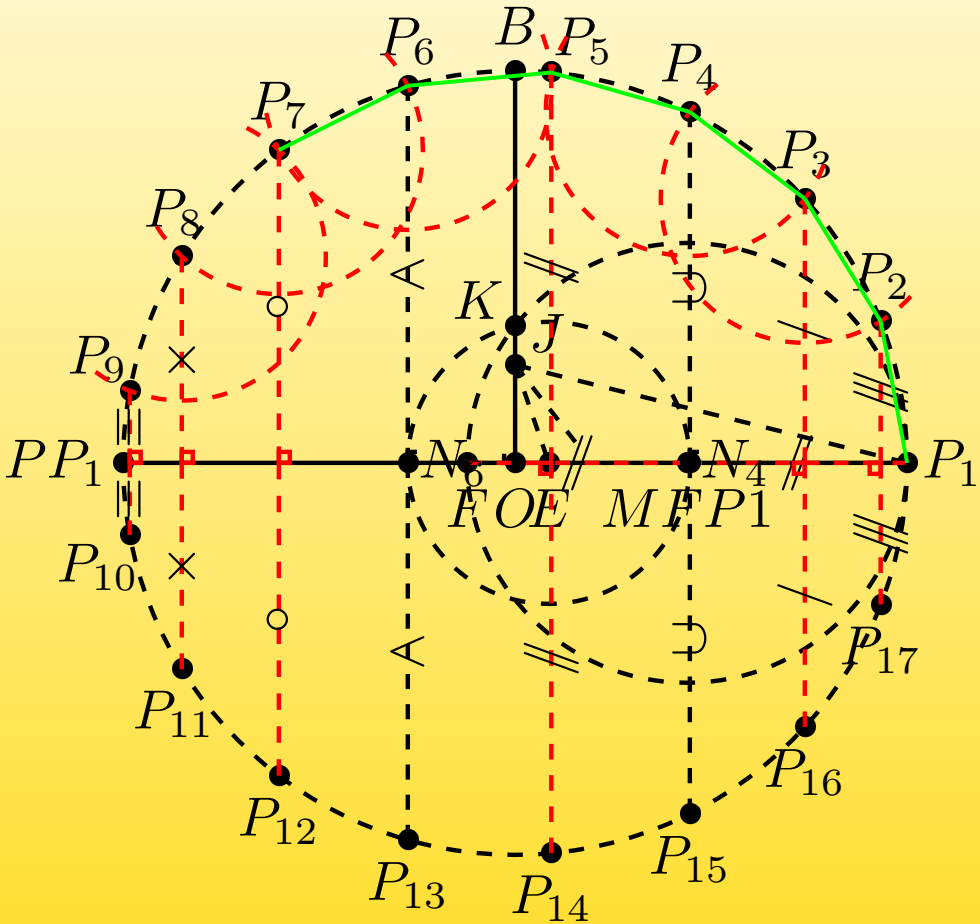


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

37: Side number 7 of the polygon

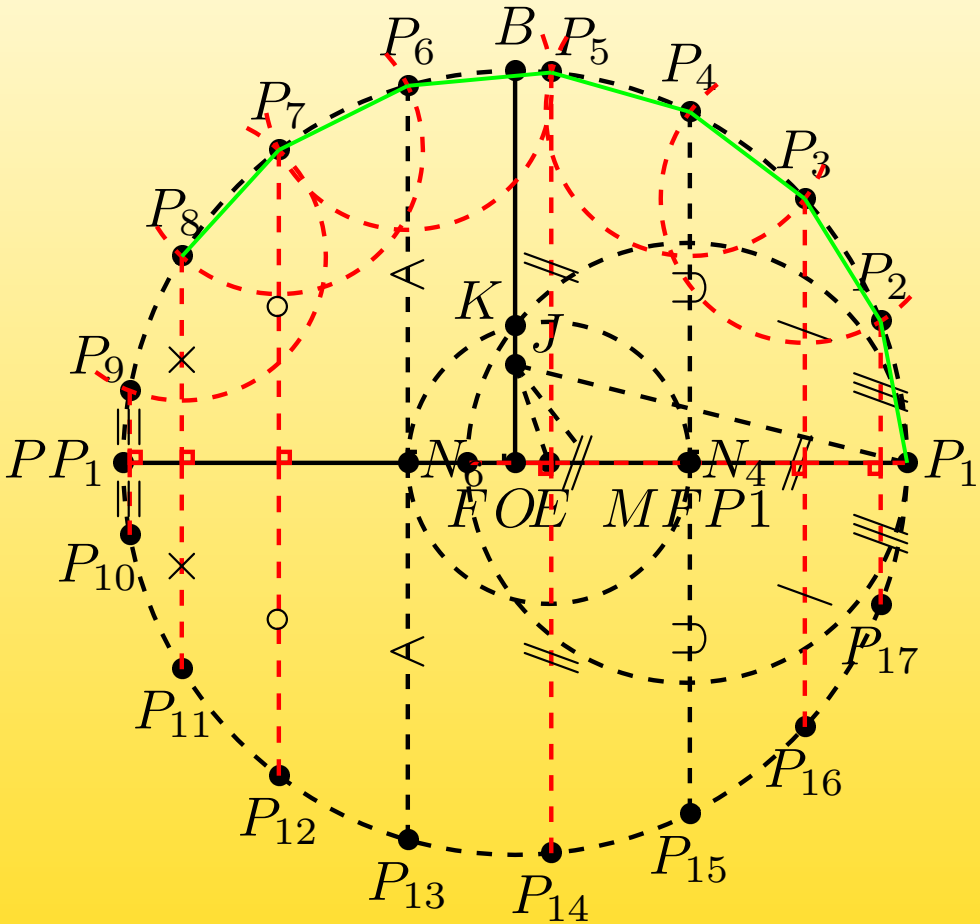


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

38: Side number 8 of the polygon

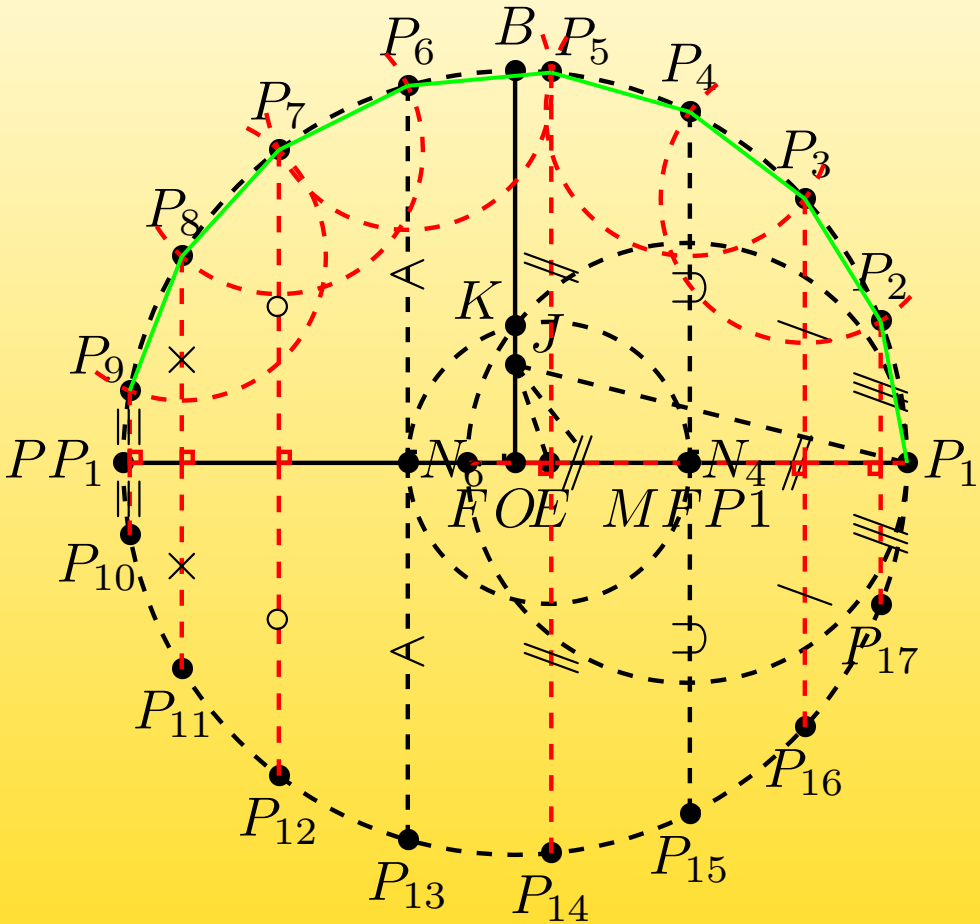


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

39: Side number 9 of the polygon

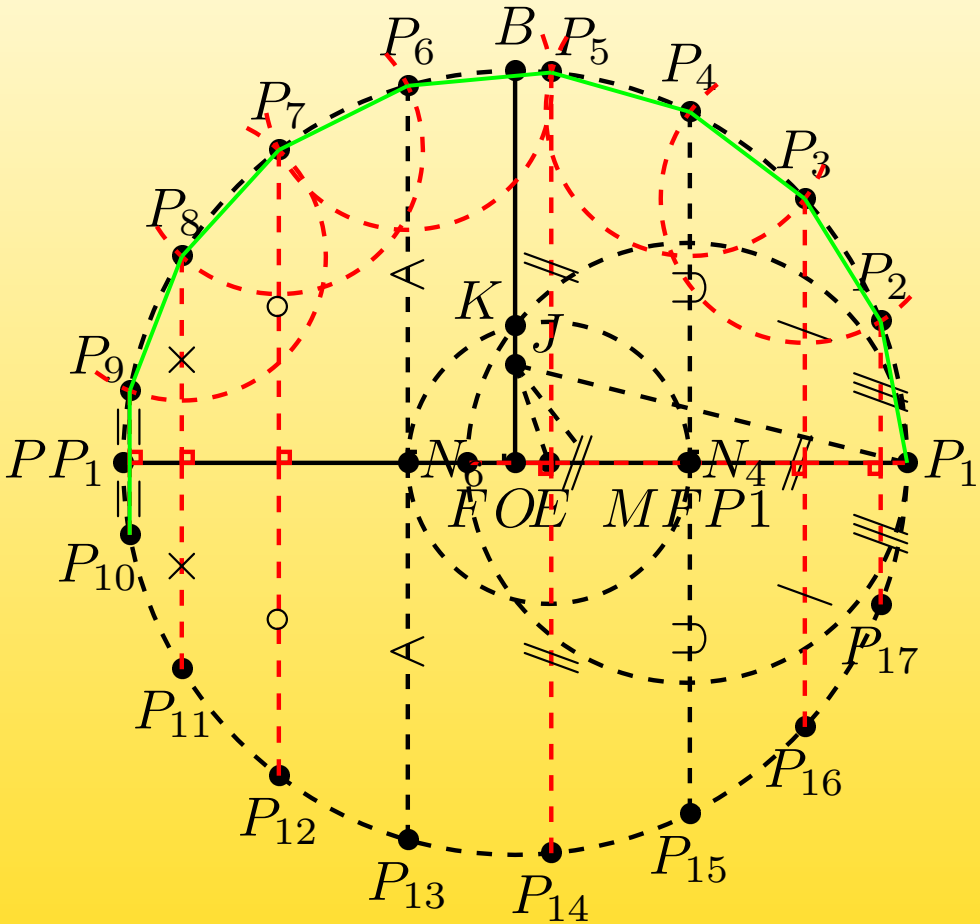


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

40: Side number 10 of the polygon

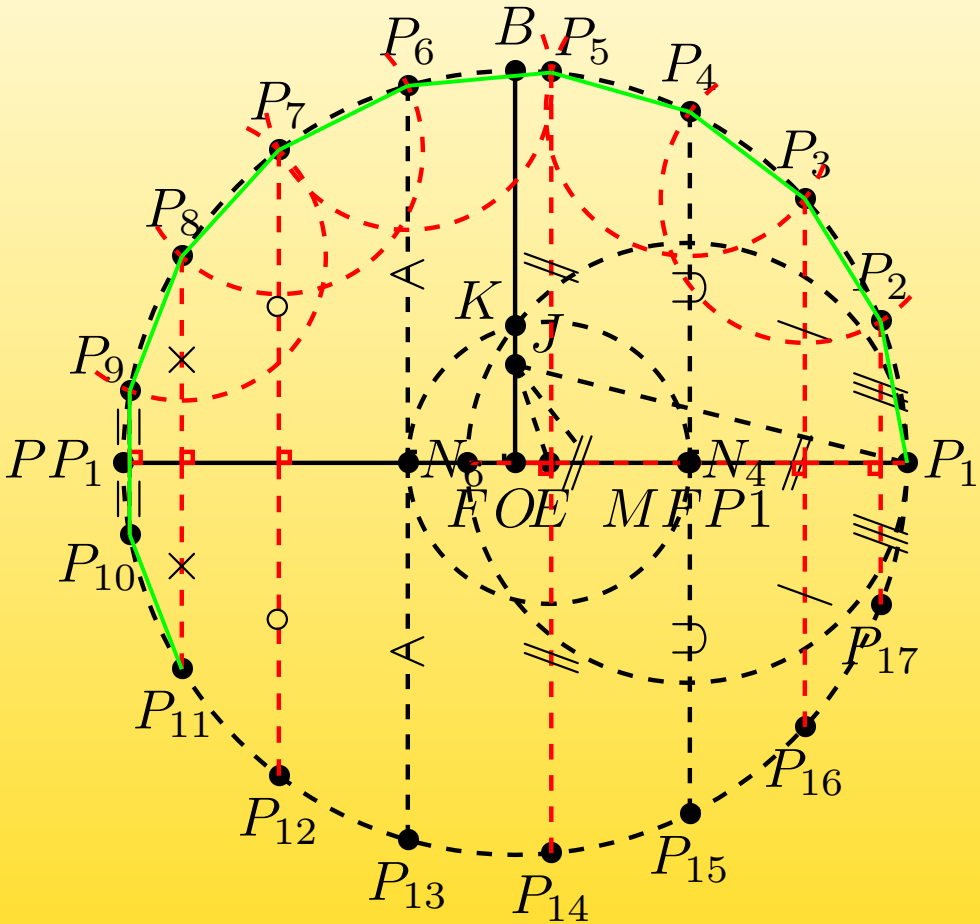


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

41: Side number 11 of the polygon

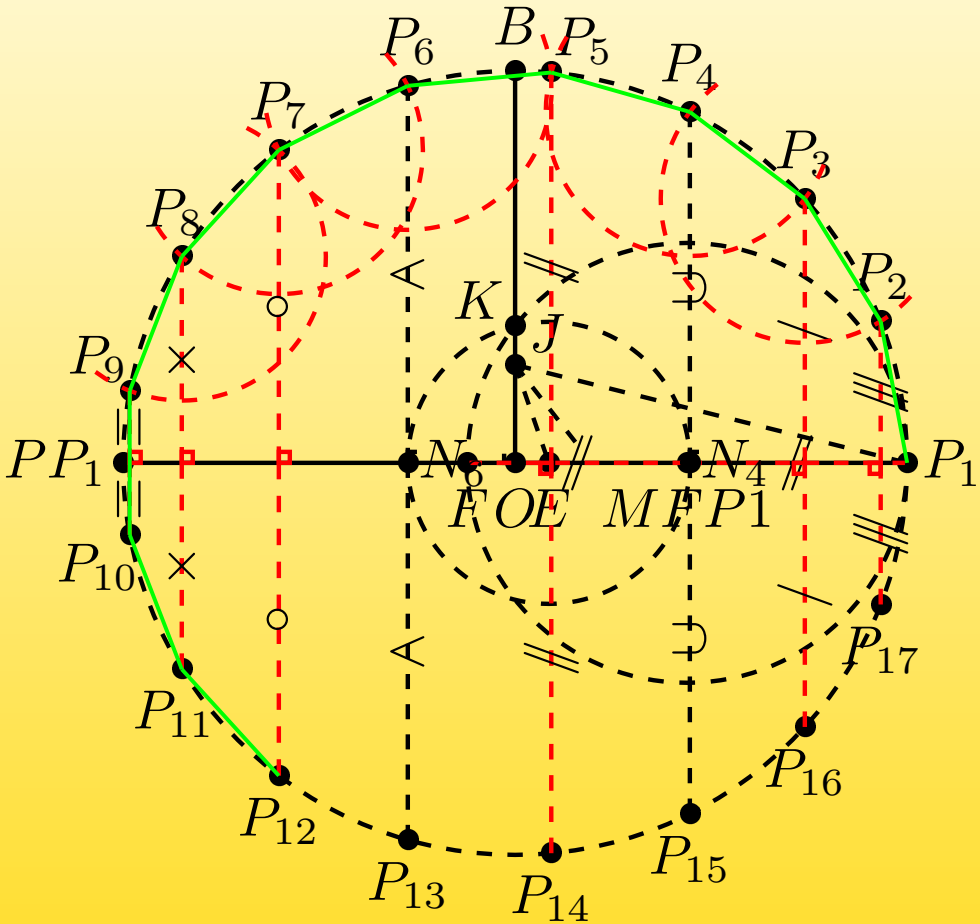


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

42: Side number 12 of the polygon

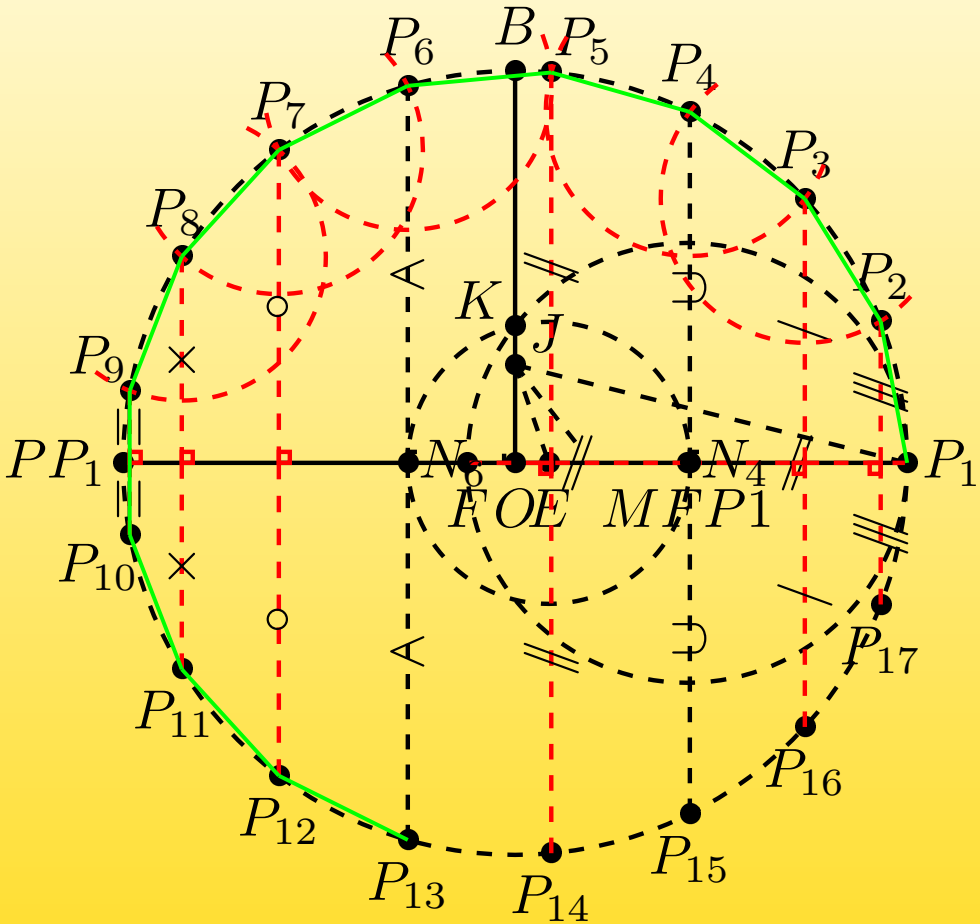


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

43: Side number 13 of the polygon

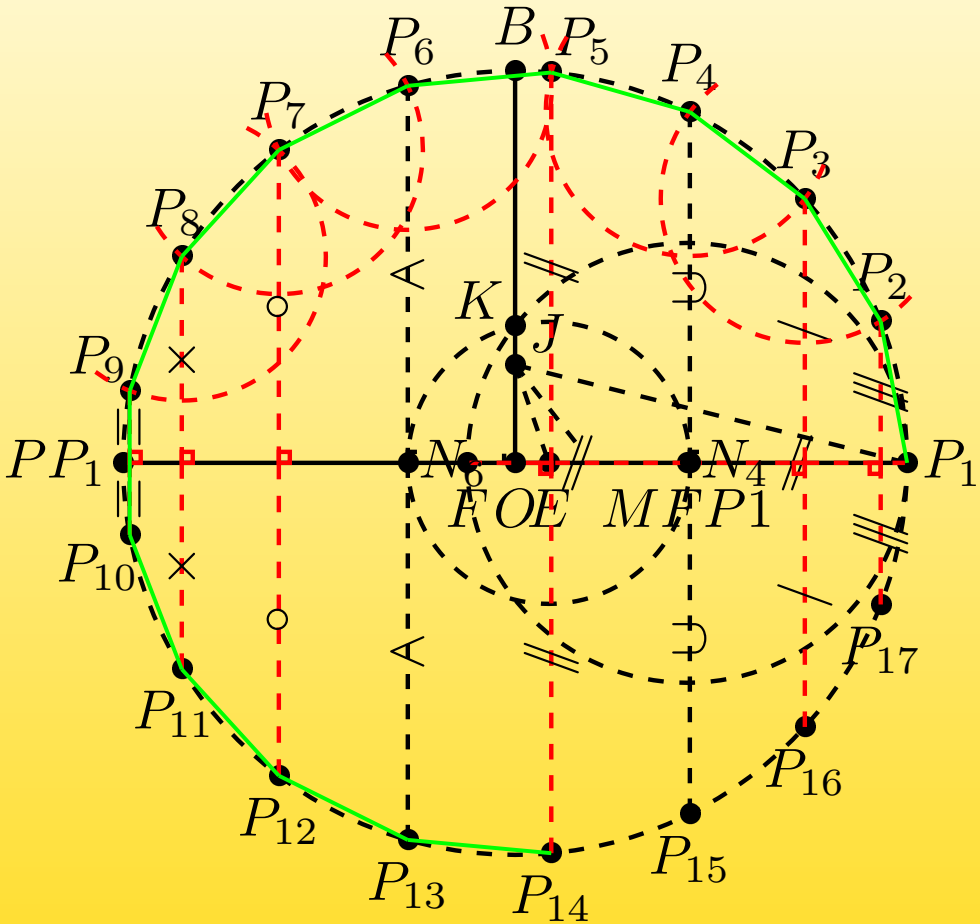


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

44: Side number 14 of the polygon

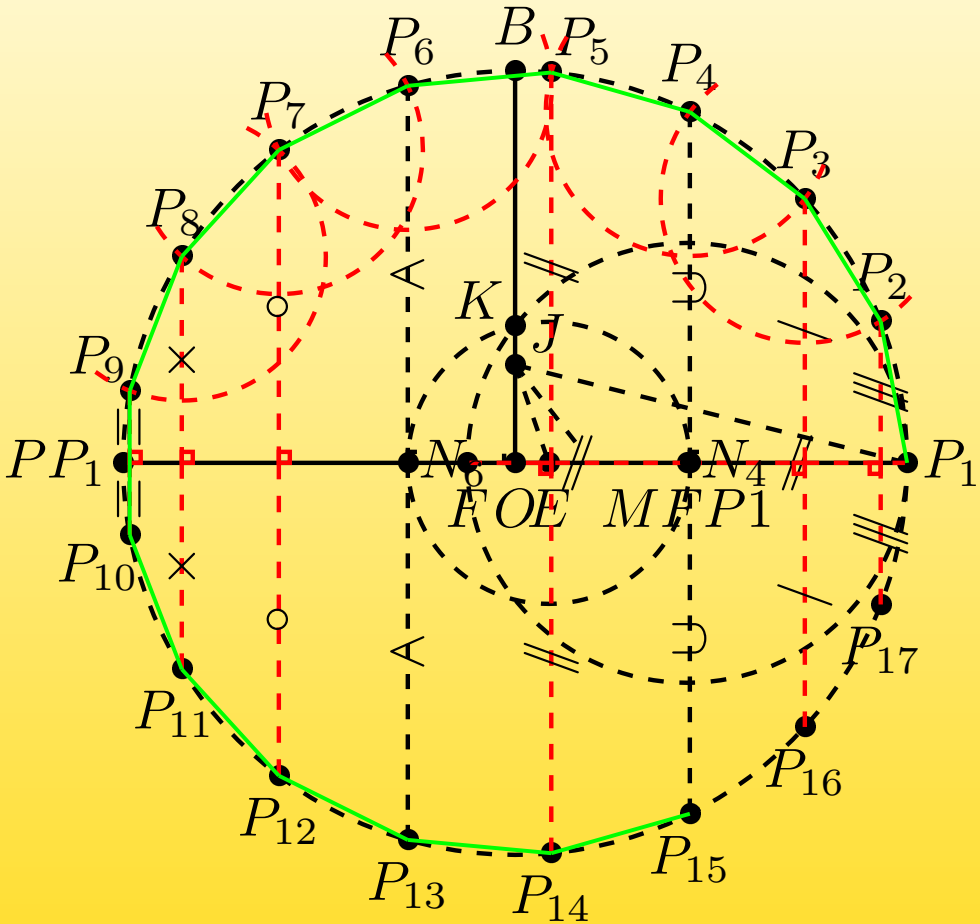


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

45: Side number 15 of the polygon

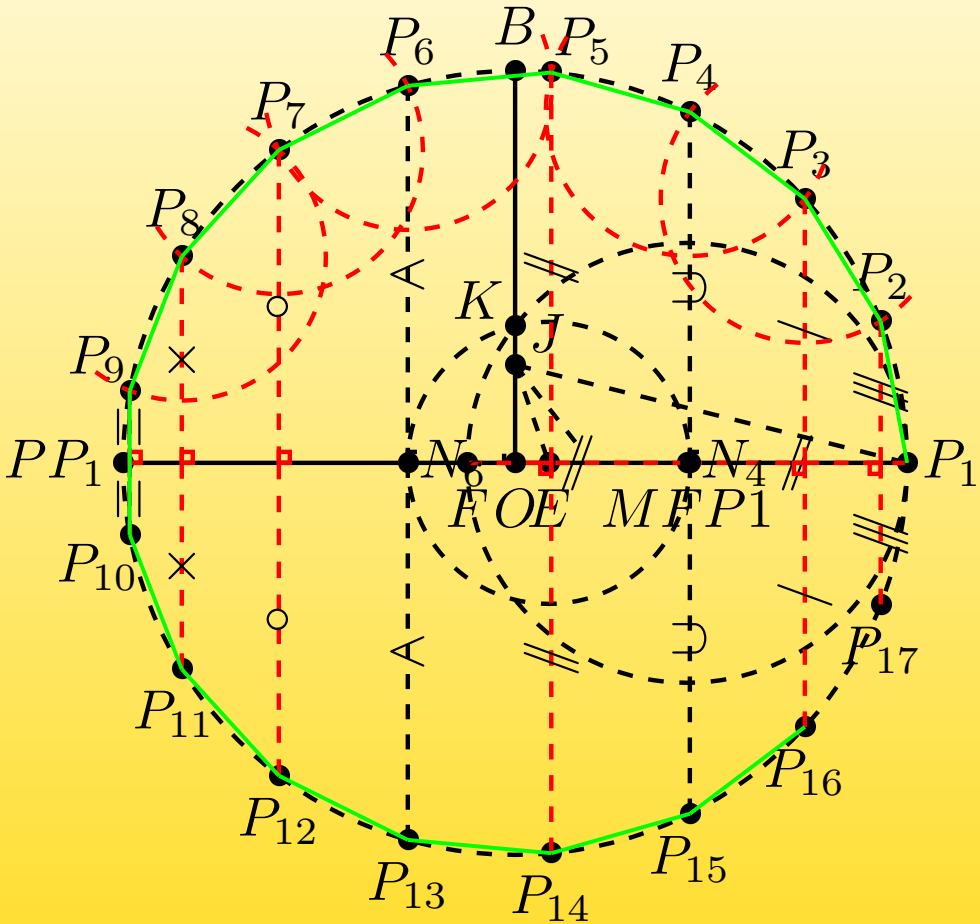


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

46: Side number 16 of the polygon

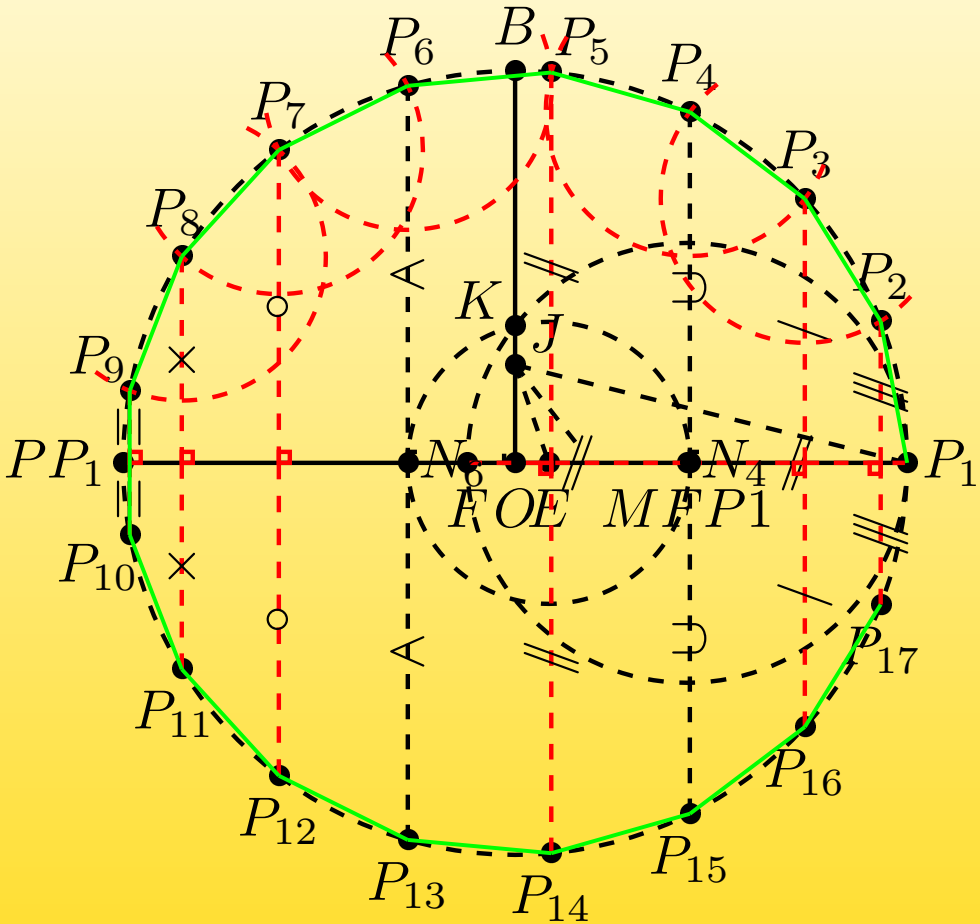


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

11 – Building of a regular polygon of seventeen sides

47: Side number 17 of the polygon

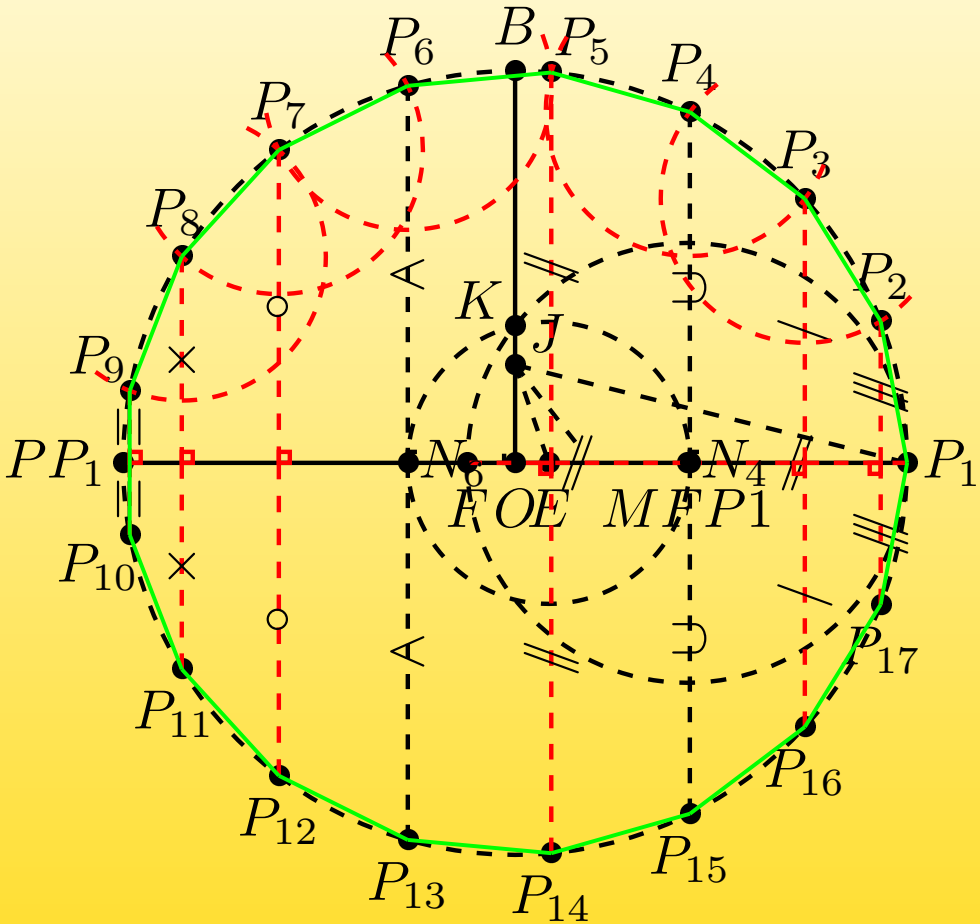


Figure 6: Building of a regular polygon of seventeen sides, according to the method introduced by Carl Friedrich GAUSS.
This code is from Dominique RODRIGUEZ, part of the examples of his 'pst-eucl' PSTricks package for Euclidian geometry.

End of animation

12 – External files inclusion

Figure 7: External files inclusion

End of animation

12 – External files inclusion

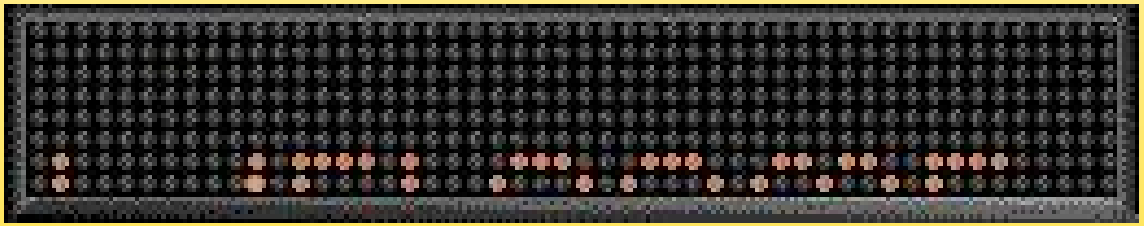


Figure 7: External files inclusion

End of animation

12 – External files inclusion

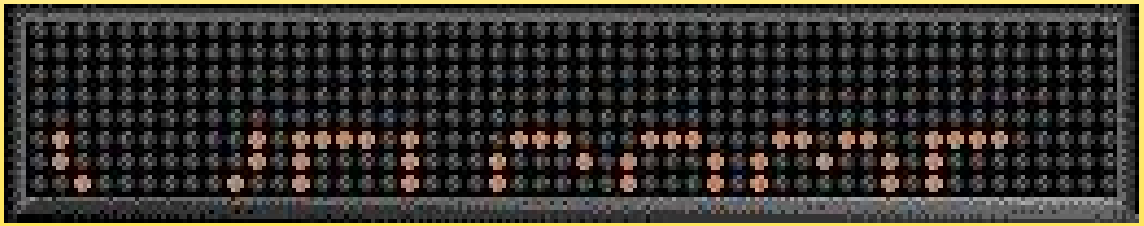


Figure 7: External files inclusion

End of animation

12 – External files inclusion

Figure 7: External files inclusion

End of animation

12 – External files inclusion

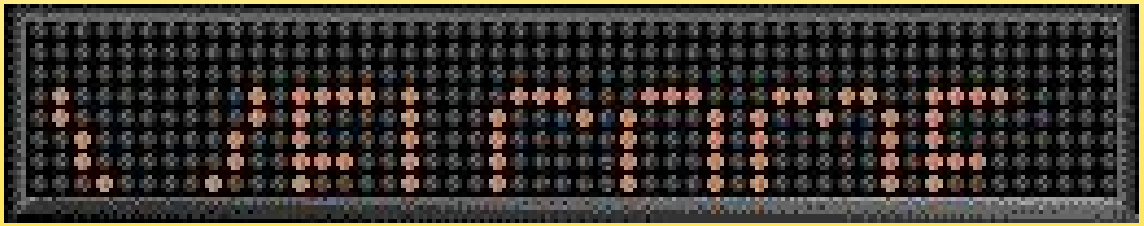


Figure 7: External files inclusion

End of animation

12 – External files inclusion

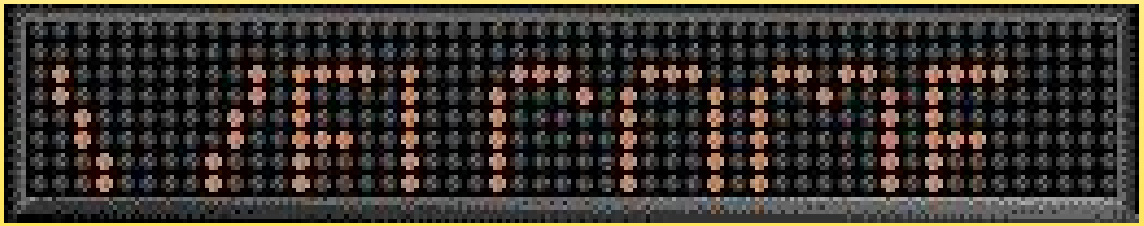


Figure 7: External files inclusion

End of animation

12 – External files inclusion

Figure 7: External files inclusion

End of animation

12 – External files inclusion

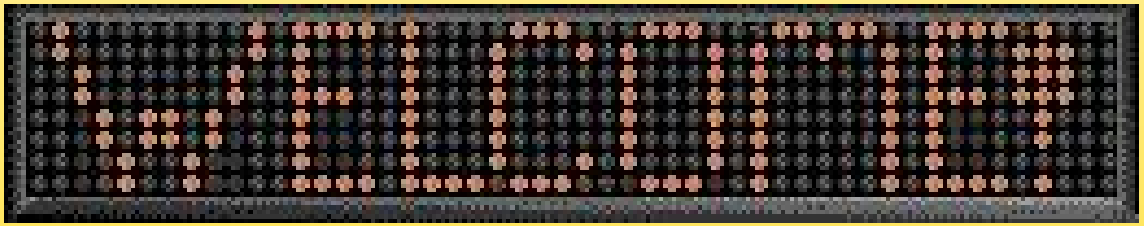


Figure 7: External files inclusion

End of animation

12 – External files inclusion

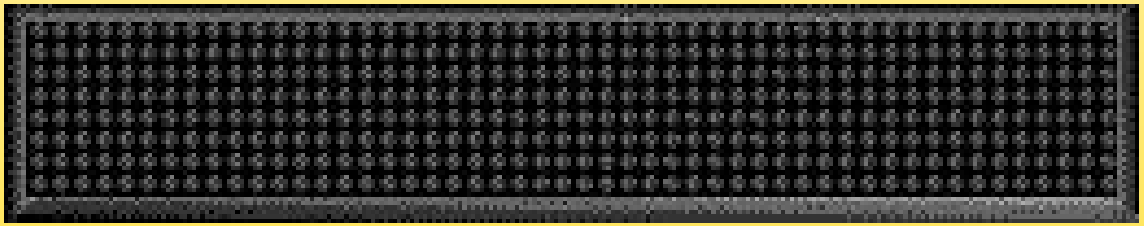


Figure 7: External files inclusion

End of animation

12 – External files inclusion

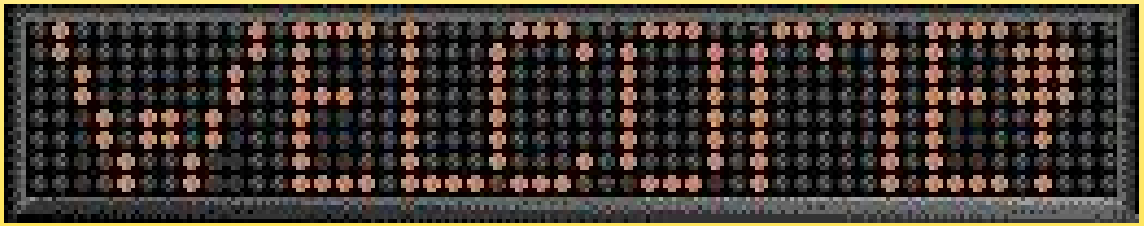


Figure 7: External files inclusion

End of animation

12 – External files inclusion

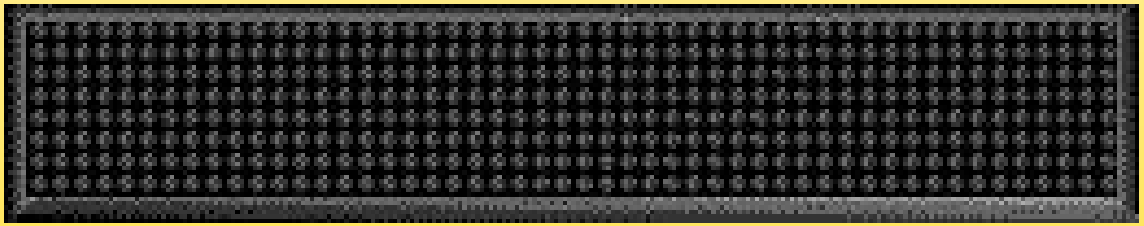


Figure 7: External files inclusion

End of animation

12 – External files inclusion

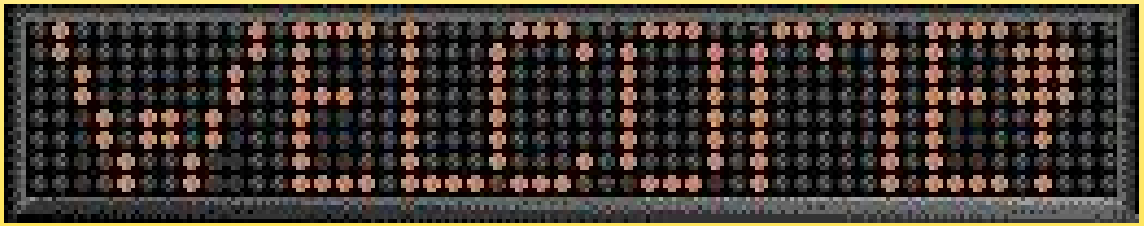


Figure 7: External files inclusion

End of animation

12 – External files inclusion

Figure 7: External files inclusion

End of animation

12 – External files inclusion

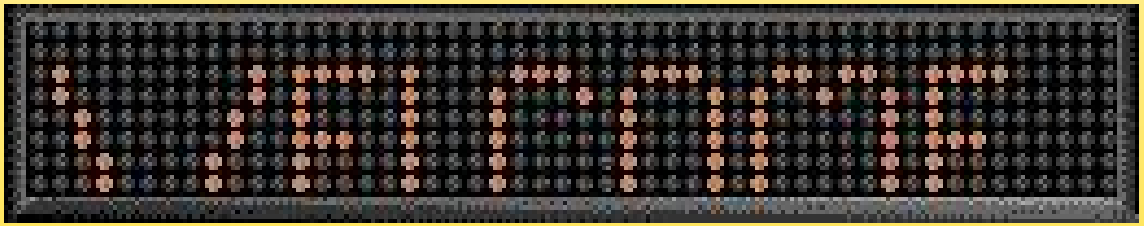


Figure 7: External files inclusion

End of animation

12 – External files inclusion

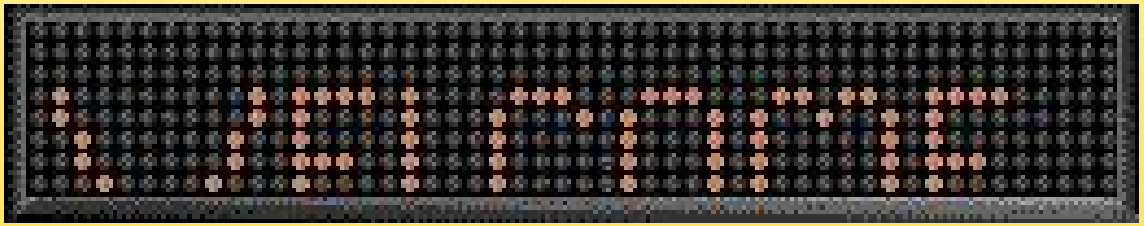


Figure 7: External files inclusion

End of animation

12 – External files inclusion

Figure 7: External files inclusion

End of animation

12 – External files inclusion

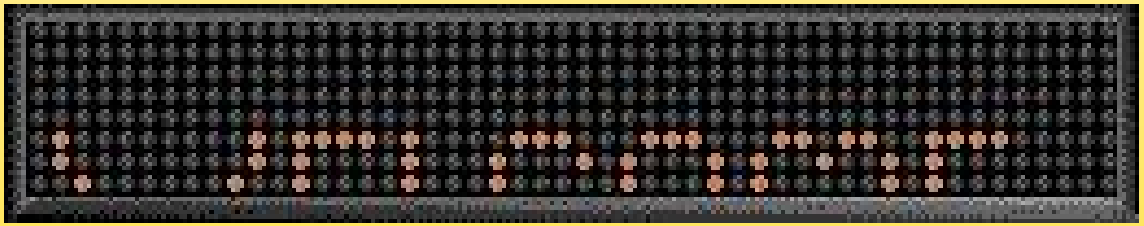


Figure 7: External files inclusion

End of animation

12 – External files inclusion

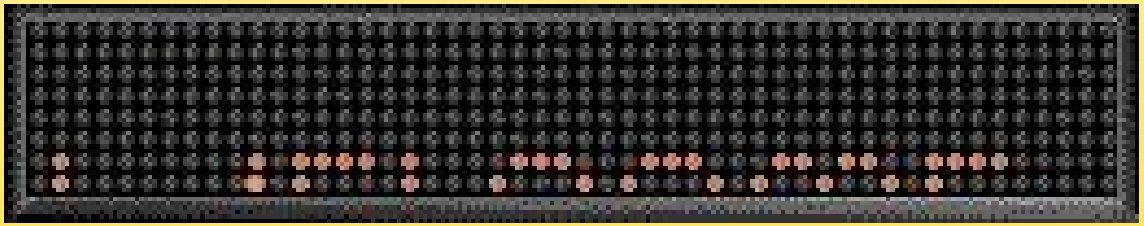


Figure 7: External files inclusion

End of animation

12 – External files inclusion

Figure 7: External files inclusion

End of animation

12 – External files inclusion

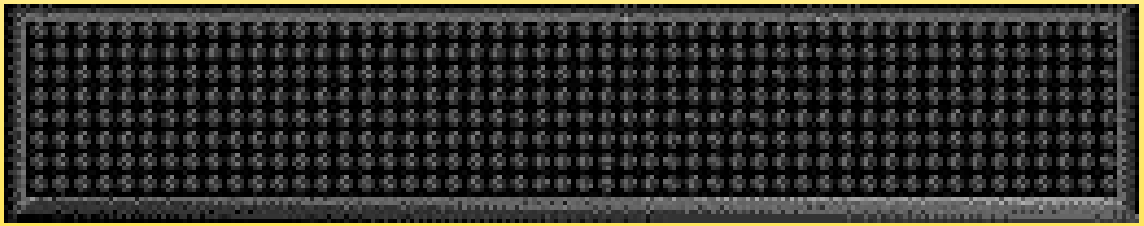


Figure 7: External files inclusion

End of animation