The CGAL PostScript Stream

Franck Descollonges
INRIA Sophia Antipolis

29/01/1997

1 Introduction

Postscript is a page description language. It is a programming language designed to convey a description of virtually any desired page to a printer.

We provide a Postscript output for all geometric classes of CGAL. Thanks to this stream, you will be able to create file printable or a file you will be able to insert in Latex documents. Indeed the generate output file can be an EPSF file.

2 The Postscript Stream Class

This class is the Postscript Stream class. It is the one containing all methods used to create, modify a Postscript output. The following two macros contain the public and the private part respectively.

class CGAL_PS_Stream[1] $+\equiv$

```
typedef const char *DashStyle;

class CGAL_PS_Stream {
public:

  typedef CGAL_Bbox_2 PS_BBox;

  static const DashStyle SOLID;
  static const DashStyle DASH1;
  static const DashStyle DASH2;
  static const DashStyle DASH3;
  static const DashStyle DASH4;
  static const DashStyle DASH5;
  static const DashStyle DASH6;
  static const float CM;
  static const float INCH;
  static const float POINT;

  enum OutputMode {READABLE, QUIET, READABLE_EPS, QUIET_EPS, GS_VIEW};
  enum DotStyle {NONE, XCROSS, ICROSS, EDOT, FDOT, EBOX, FBOX};
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.1 The Graphical Environment : Auxiliary classes

Before speacking about the Postscript stream, we have to introduce some auxiliary classes.

2.1.1 The Axis Class

If you want to represente axis on the postscript output, you have to create an axis object. You can set the step between two marks on the x-axis and the y-axis. If the padx or the pady is 0 no mark are displayed on the axis. The thick member contains thickness of the lines rrepresenting the axis. You can insert only one axis object in a postscript stream.

class CGAL_PS_Stream[2] + ≡

```
class Axis {

friend CGAL_PS_Stream;
friend CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const Axis &g);

public:
  Axis(double x, double y, unsigned int t=0.0)
    : _stepx(x), _stepy(y), _thick(t) {}
  Axis(double xy, unsigned int t=0.0)
    : _stepx(xy),_stepy(xy), _thick(t) {}
  Axis(): _stepx(1.0), _stepy(1.0), _thick(0.0) {}

  double stepx()     const { return _stepx;}
  double stepy()     const { return _stepy;}
  double thickness() const { return _thick;}
private:
  double _stepx;
  double _stepy;
  bool _thick;
};
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.1.2 The Grid Class

You can insert grids in a postscript stream. You have to create an object and select the step beetween two line for the x-axis and the y-axis... You can select the style of the mark and insert many grid of differents styles.

class CGAL_PS_Stream[3] + ≡

```
class Grid {

friend CGAL_PS_Stream;
friend CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const Grid &g);

public:
  Grid(double x, double y, DashStyle str="[1 5] 0")
    : _stepx(x), _stepy(y) {_style=strdup(str);}
  Grid(double xy, DashStyle str="[1 5] 0")
```

```
      : _stepx(xy), _stepy(xy) {_style=strdup(str);}
    Grid() : _stepx(1.0), _stepy(1.0), _style("[1 5] 0") {}

  private:
    double    stepx() const { return _stepx;}
    double    stepy() const { return _stepy;}
    DashStyle style() const { return _style;}

    double _stepx;
    double _stepy;
    DashStyle _style;
  };
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.1.3 The Label class

The Label class is used to insert text in a postscript output. You have just to define a Label object and insert it in the stream to display the string in the postscript output. Default font and size is used to display texts but you are able to change this using the modifiers of the stream.

class CGAL_PS_Stream[4] + ≡

```
  class Label {
  friend CGAL_PS_Stream;
    friend CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const Label &txt);
  public:
    Label(const char* txt) { _text=strdup(txt);}
  private:
    Label() { _text="";}
    const char* text() const { return _text;}
    const char* _text;
  };
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.1.4 The Latex label class

These class is used to insert text which will be only display with Latex. These labels used default latex font to be displayed. You can can change font and style (for exemple math style) by using standart latex commands as string like :

```
{$\alpha^{y}_{i}$}
```

These labels wont be displayed on a gs view but only with latex. To include such a file you have to type ("toto.ps" is the file in which the Postscript output of CGAL had been redirected) :

```
\documentclass[]{article}
\begin{document}
\def\Ipe#1{\def\IPEfile{#1}\input{#1}}
```

```
      before.

\begin{figure*}[h]
\begin{center}
\Ipe{toto.ps}
\end{center}
\end{figure*}

      after.

\end{document}
```

class CGAL_PS_Stream[5] + ≡

```
    class Latex_Label {
      friend CGAL_PS_Stream;
      friend CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, Latex_Label &txt);
    public:
      // Only text can be define by user
      Latex_Label(const char* txt) { _text=strdup(txt);posx=0;posy=0;}
      Latex_Label() { _text="";posx=0;posy=0;}
    private:
      // These functions are private because the position of the string must never
appears to the user
      // Only stream modifiers will access these data
      void setposition(float x, float y) { posx=x;posy=y;}
      Latex_Label(const char* txt,float x, float y)
{_text=strdup(txt);posx=x;posy=y;}

      float xpos() const {return posx;};
      float ypos() const {return posy;};
      const char* text() const { return _text;};

      // Slots
      const char* _text;
      float posx;
      float posy;
    };
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

I need to insert all Latex labels at the same position (at the end of the file.) So i need to save all data in a list. It is why i define a Lsit of latex label.

class CGAL_PS_Stream[6] + ≡

```
    typedef list<Latex_Label> List_Label;
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.1.5 The Border class

The border class is used to insert a border in a stream. Users just need to define a border, and to insert it in the stream. The postscript output will display this border.

class CGAL_PS_Stream[7] + ≡

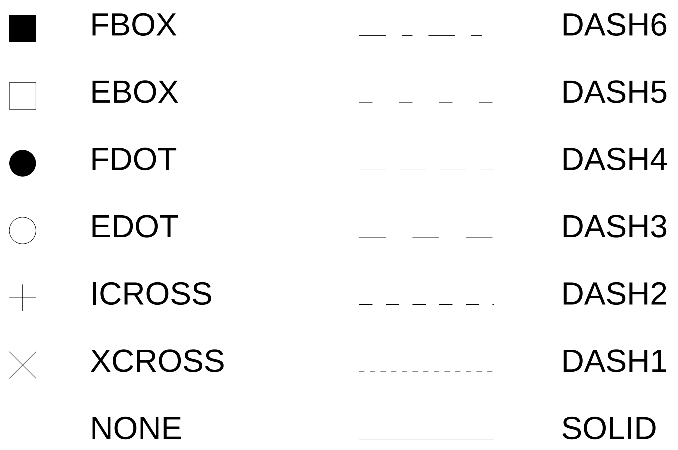
```
class Border {
  friend CGAL_PS_Stream;
  friend CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const Border &b);
public:
  Border(int s=0) { _size=s;}
private:
  int size() const { return _size;}
  int _size;
};
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.1.6 The Context class

The Context class define a graphical context. It is used to select the style of the drawing.

You can create a context (set to DEFAULT value) and copy the current context. You only can change the current context of the postscript stream by inserting modifiers in the stream. The context can be globaly set with a modifier in the Postscript stream thanks to this class.



The following two macros contain the public and the private part respectively.

The Public Interface of the Context class

class CGAL_PS_Stream[8] + ≡

```
class Context {
```

```
    friend CGAL_PS_Stream;

    public:

    Context() : _border_color(CGAL_Color(0,0,0)),
      _fill_color(CGAL_Color(0,0,0)),_dot_style(XCROSS),_dot_size(5),
      _thickness(0),_line_style(SOLID),_fill(false),
      _font("Helvetica"),_font_size(12), _anchor_point(CGAL_Point_2<CGAL_Cartesian
<double> > (0,0))
      {}

    Context(const Context& c)
      :  _border_color(c.get_border_color()),_fill_color(c.get_fill_color()),
         _dot_style(c.get_dot_style()), _dot_size(c.get_dot_size()),
         _thickness(c.get_thickness()), _line_style(strdup(c.get_line_style())),
         _fill(c.get_fill()), _font(strdup(c.get_font())),
_font_size(c.get_font_size()),
         _anchor_point(c.get_pos())
      {};

    // Accessor

    CGAL_Color   get_border_color()  const {return _border_color;}
    CGAL_Color   get_fill_color() const {return _fill_color;}
    DotStyle     get_dot_style() const {return _dot_style;}
    unsigned int get_dot_size()  const {return _dot_size;}
    unsigned int get_thickness()     const {return _thickness;}
    unsigned int get_font_size()     const {return _font_size;}
    DashStyle    get_line_style()    const {return _line_style;}
    const char*  get_font()          const {return _font;}
    bool         get_fill()          const {return _fill;}
    CGAL_Point_2<CGAL_Cartesian <double> > get_pos() const {return _anchor_point;}

    void set_border_color(CGAL_Color& c) {_border_color=c;}
    void set_fill_color(CGAL_Color& c)   {_fill_color=c;}
    void set_dot_style(DotStyle& s)    {_dot_style=s;}
    void set_dot_size(unsigned int s)     {_dot_size=s;}
    void set_thickness(unsigned int t)    {_thickness=t;}
    void set_font_size(unsigned int s)    {_font_size=s;}
    void set_fill(bool&b)     {_fill=b;}
    void set_current_pos(const CGAL_Point_2<CGAL_Cartesian <double> >& p)
{_anchor_point=p;}
    void set_line_style(DashStyle style) {_line_style=strdup(style);}
    void set_font(const char *font) {_font=strdup(font);}
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

The Private part of the Context class

class CGAL_PS_Stream[9] + ≡

```
    private:
```

```
    // Store the current border color
    CGAL_Color _border_color;
    // Store the current fill color
    CGAL_Color _fill_color;
    // Store the current dot style
    enum DotStyle _dot_style;
    // Store the current dot size
    unsigned int _dot_size;
    // Store the current line thickness
    unsigned int _thickness;
    // Store the current line style
    DashStyle _line_style;
    // Define if a polygone must be fill or not
    bool _fill;
    // Define if direction must be shown.
    bool _dir;
    // Store the name of the font to use. It is only used  for standart Label, not
for Latex Label.
    const char* _font;
    // Store the size of the font. It is only used  for standart Label, not for
Latex Label.
    unsigned int _font_size;
    //Anchor point:
    CGAL_Point_2<CGAL_Cartesian <double> > _anchor_point;

    };
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.2 Specification of the CGAL_PS_Stream class

2.2.1 The public part of CGAL_PS_Stream

The different constructors of the Postscipt Stream allow a very easy definition of the working area. For all constructors, you need to define a bounding box. This is the maximum area of real objects represented in the postscript view.

In this coonstructors, the whole page is used, the bounding box is defined by bb.

class CGAL_PS_Stream[10] + ≡

```
      CGAL_PS_Stream(const PS_BBox &bb, ostream &os,
                     OutputMode = QUIET);
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

The same as above but the output stream will be a file which name is fname.

class CGAL_PS_Stream[11] + ≡

```
          CGAL_PS_Stream(const PS_BBox &bb, const char *fname,
                         OutputMode = QUIET);
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

These constructors specify the size of the postscript result. (size is given in postscript dots.) You can select the desired height of the output. The width of the output will be calculated to keep circle round.

In an output stream :

class CGAL_PS_Stream[12] + ≡

```
CGAL_PS_Stream(const PS_BBox &bb,float H, ostream &os,
               OutputMode = QUIET);
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

In a file named fname :

class CGAL_PS_Stream[13] + ≡

```
CGAL_PS_Stream(const PS_BBox &bb,float H, const char *fname,
               OutputMode = QUIET);
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

These constructors specify the size of the postscript result in postscipt dots.

In an output stream :

class CGAL_PS_Stream[14] + ≡

```
CGAL_PS_Stream(const PS_BBox &bb,float L, float H, ostream &os,
               OutputMode = QUIET);
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

In a file named fname :

class CGAL_PS_Stream[15] + ≡

```
CGAL_PS_Stream(const PS_BBox &bb,float L, float H, const char *fname,
               OutputMode = QUIET);
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

Destructor :

class CGAL_PS_Stream[16] + ≡

```
~CGAL_PS_Stream();
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

//Manipulators

class CGAL_PS_Stream[17] + ≡

```
CGAL_PS_Stream& _SetBorderColor(CGAL_Color &);
CGAL_PS_Stream& _SetFillColor(CGAL_Color &);
CGAL_PS_Stream& _SetPointSize(unsigned int);
```

```
CGAL_PS_Stream& _SetLineWidth(unsigned int);
CGAL_PS_Stream& _SetPointStyle(enum DotStyle);
CGAL_PS_Stream& _SetLineStyle(DashStyle);
CGAL_PS_Stream& _SetFill(bool);
CGAL_PS_Stream& _SetDefaultContext(void);
CGAL_PS_Stream& _SetCurrentContext(const Context &);
CGAL_PS_Stream& _ShowDirection(bool);
CGAL_PS_Stream& _MoveTo(CGAL_Point_2< CGAL_Cartesian <double> >);
CGAL_PS_Stream& _ShowAxis(Axis &);
CGAL_PS_Stream& _ShowGrid(Grid &);
CGAL_PS_Stream& _PutPsLabel(const char *);
CGAL_PS_Stream& _PutLatexLabel(const char *);
CGAL_PS_Stream& _PutBorder(unsigned int);
CGAL_PS_Stream& _SetFont(const char *);
CGAL_PS_Stream& _SetFontSize(unsigned int);


//Accessors
ostream&        os()   {return _os;}
List_Label&   list()        {return _ll;}
const Context context() const {return ctxt;}
bool         gs_output()     const {return (bool)(mode()==GS_VIEW);}
PS_BBox      bbox()          const {return _bbox;}
int          width()         const {return _width;}
int          height()        const {return _height;}
OutputMode   mode()          const {return _mode;}

// Utils
double xratio() { return _xratio;}
double yratio() { return _yratio;}
double x2ps(double x) {
  return (x-_bbox.xmin())*xratio();
}
double y2ps(double y) {
  return (y-_bbox.ymin())*yratio();

}

bool is_eps();
bool is_readable();
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.2.2 The Private Part of CGAL_PS_Stream

class CGAL_PS_Stream[18] + ≡

```
private:
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

Constructors in a pipe :

class CGAL_PS_Stream[19] + ≡
```
    CGAL_PS_Stream(const PS_BBox &bb);

    CGAL_PS_Stream(const PS_BBox &bb,float H);

    CGAL_PS_Stream(const PS_BBox &bb,float L, float H);
```
This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

Default Constructor protected.

class CGAL_PS_Stream[20] + ≡
```
    CGAL_PS_Stream();

    // Manipulation du contexte.
    void setdefault();
    void setcontext();

    // Pour inserer l'entete
    void insert_catalogue();

    // Define the scale.
    double _xratio;
    double _yratio;
    // Define the boounding box
    PS_BBox _bbox;
    // OutputMode
    OutputMode _mode;
    // Size of output.
    int _width,_height;
    // Graphical Context
    Context ctxt;
    // The Output stream
    #ifdef CGAL_WORKAROUND_016
    _IO_ostream_withassign &_os;
    #else
    ostream_withassign &_os;
    #endif

    //List of Latex Labels. They will be inserted at the end of the file.
    List_Label _ll;
    };
```
This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

Definition of external fonctions and variables

class CGAL_PS_Stream[21] + ≡
```
    extern const CGAL_PS_Stream::Context CGAL_CTXT_DEFAULT;

    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Border
&b);

    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Axis &g);
```

```
    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Grid &g);

    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Label
&txt);

    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const
CGAL_PS_Stream::Latex_Label &txt);

    extern CGAL_PS_Modifier_creator<CGAL_Color &> set_fill_color;

    extern CGAL_PS_Modifier_creator<CGAL_Color &> set_border_color;

    extern CGAL_PS_Modifier_creator<unsigned int> set_point_size;

    extern CGAL_PS_Modifier_creator<CGAL_PS_Stream::DotStyle> set_point_style;

    extern CGAL_PS_Modifier_creator<DashStyle> set_line_style;

    extern CGAL_PS_Modifier_creator<unsigned int> set_line_width;

    extern CGAL_PS_Modifier_creator<bool> set_fill;

    extern CGAL_PS_Modifier_creator<const CGAL_PS_Stream::Context &>
set_current_context;

    extern CGAL_PS_Modifier_creator<bool> show_direction;

    extern CGAL_PS_Modifier_creator<CGAL_Point_2< CGAL_Cartesian <double> > >
move_to;

    //extern CGAL_PS_Modifier_creator<CGAL_PS_Stream::Axis &> show_axis;

    //extern CGAL_PS_Modifier_creator<CGAL_PS_Stream::Grid &> show_grid;

    //extern CGAL_PS_Modifier_creator<const char *> put_ps_label;

    //extern CGAL_PS_Modifier_creator<const char *> put_latex_label;

    //extern CGAL_PS_Modifier_creator<unsigned int> put_border;

    extern CGAL_PS_Modifier_creator<const char *> set_font;

    extern CGAL_PS_Modifier_creator<unsigned int> set_font_size;
```

This macro is defined in definitions 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, and 21.
This macro is invoked in definition 41.

2.3 Implementation : Constructors and Destructors

Definition of constants :

Constructors and Destructors[22] $+\equiv$

```
    const float CGAL_PS_Stream::CM=28.4;
```

```
    const float CGAL_PS_Stream::INCH=72.0;
    const float CGAL_PS_Stream::POINT=1.0;

    const DashStyle CGAL_PS_Stream::SOLID="[] 0 ";
    const DashStyle CGAL_PS_Stream::DASH2="[5 5] 0 ";
    const DashStyle CGAL_PS_Stream::DASH3="[10 10] 0 ";
    const DashStyle CGAL_PS_Stream::DASH6="[10 6 4 6] 0 ";
    const DashStyle CGAL_PS_Stream::DASH4="[10 5] 0 ";
    const DashStyle CGAL_PS_Stream::DASH5="[5 10] 0 ";
    const DashStyle CGAL_PS_Stream::DASH1="[2 2] 0 ";
    extern const CGAL_PS_Stream::Context
CGAL_CTXT_DEFAULT=CGAL_PS_Stream::Context();
```

This macro is defined in definitions 22, 23, 24, and 25.
This macro is invoked in definition 40.

Constructors :

Constructors and Destructors[23] + ≡

```
    /*
      CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb)
      : _bbox(bb), _os(cerr), _mode(GS_VIEW)
      {
      FILE *fp = popen("gs -","w");
      if(!fp){
      cerr « "Could not open pipe to gs" « endl ;
      exit(-1);
      }
      os().attach(fileno(fp));
      insert_catalogue();
      }
      CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb,float L, float H)
      : _bbox(bb), _os(cerr), _mode(GS_VIEW), _width(L), _height(H)
      {
      insert_catalogue();
      _xratio=_width/(_bbox.xmax()-_bbox.xmin());
      _yratio=_height/(_bbox.ymax()-_bbox.ymin());
      };

    */
```

This macro is defined in definitions 22, 23, 24, and 25.
This macro is invoked in definition 40.

Alternatively you can pass an output stream as argument. The formatted Postscript output goes to this stream then, which typically will be `cout` or a file (passing a `fstream`).

Constructors and Destructors[24] + ≡

```
    CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb, ostream &os,
                                   OutputMode mode)
      :_bbox(bb), _mode(mode),
      _width((int)21*CM), _height((int)29.7*CM), _os(cerr)
    {
      _xratio=_width/(_bbox.xmax()-_bbox.xmin());
```

```
  _yratio=_height/(_bbox.ymax()-_bbox.ymin());
  _os=os;
  insert_catalogue();
}

CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb, const char *fname,
                               OutputMode mode)
  : _bbox(bb),_mode(mode),_width((int)21*CM), _height((int)29.7*CM),_os(cerr)
{
  _xratio=_width/(_bbox.xmax()-_bbox.xmin());
  _yratio=_height/(_bbox.ymax()-_bbox.ymin());
  static ofstream os(fname);
  _os=os;
  insert_catalogue();
}

CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb,float H, ostream &os,
                               OutputMode mode)
  : _bbox(bb), _mode(mode), _height(H), _os(cerr)
{
  _width=(bb.xmax()-bb.xmin())*H/(bb.ymax()-bb.ymin());
  _xratio=_width/(_bbox.xmax()-_bbox.xmin());
  _yratio=_height/(_bbox.ymax()-_bbox.ymin());
  _os=os;
  insert_catalogue();
}

CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb,float H, const char *fname,
                               OutputMode m)
  : _bbox(bb), _mode(m), _height(H), _os(cerr)
{
  static ofstream os(fname);
  _os=os;
  _width=(bb.xmax()-bb.xmin())*H/(bb.ymax()-bb.ymin());
  _xratio=_width/(_bbox.xmax()-_bbox.xmin());
  _yratio=_height/(_bbox.ymax()-_bbox.ymin());
  insert_catalogue();
}

CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb,float L, float H,
                               ostream &os, OutputMode mode)
  : _bbox(bb), _mode(mode), _width(L), _height(H), _os(cerr)
{
  _os=os;
  _xratio=_width/(_bbox.xmax()-_bbox.xmin());
  _yratio=_height/(_bbox.ymax()-_bbox.ymin());
  insert_catalogue();
}

CGAL_PS_Stream::CGAL_PS_Stream(const PS_BBox &bb,float L, float H,
                               const char *fname, OutputMode mode)
  : _bbox(bb), _mode(mode),_width(L),_height(H), _os(cerr)
{
```

```
    static ofstream os(fname);
    _os=os;
    _xratio=_width/(_bbox.xmax()-_bbox.xmin());
    _yratio=_height/(_bbox.ymax()-_bbox.ymin());
    insert_catalogue();
  }
```

This macro is defined in definitions 22, 23, 24, and 25.
This macro is invoked in definition 40.

Destructor :

The destructor insert all the latex instruction if needed. Then, it inserts all the latex labels, and finaly it closes the stream.

Constructors and Destructors[25] + ≡

```
  CGAL_PS_Stream::~CGAL_PS_Stream()
  {
    List_Label tmp=list();
    if (!list().empty())
      {
        os() « "%%}\\makeatletter\\let\\@notdefinable\\relax" «endl;
        os() «
"%%\\def\\IPEc#1[#2]#3{\\newcommand{#1}[#2]{#3}\\ignorespaces}\\@ifundefined"
«endl;
        os() «
"%%{selectfont}{\\let\\selectfont\\relax\\def\\fontsize#1#2{}}{}\\makeatother"
«endl;
        os() « "%%\\IPEc\\IPEput[4]{\\put(0,0){\\special{psfile=\\IPEfile}}}"
«endl;
        os() « "%%\\IPEc\\IPEmp[2]{\\minipage[t]{#1bp}#2\\special{color
pop}\\endminipage}" «endl;
        os() « "%%\\IPEc\\IPEtext[1]{\\makebox(0,0)[lb]{#1\\special{color
pop}}}" «endl;
        os() « "%%\\IPEc\\IPEfs[1]{\\IPEcolfs{0 0 0}{#1}}" «endl;
        os() «
"%%\\IPEc\\IPEcolfs[2]{\\dimen0=#2pt\\fontsize{#2}{1.2\\dimen0}\\selectfont"
«endl;
        os() « "%%\\special{color push rgb #1}}" «endl;
        os() « "%%\\IPEc\\IPEsize[2]{\\unitlength1bp\\ignorespaces}" «endl;

        os() « "%%\\IPEsize{" « width() « "}{"« height() « "}" «endl;
        os() « "%%\\begin{picture}(" « width() « "," « height() « ")(0,0)"
«endl;
        os() « "%%\\IPEput{0}{0}{" « width() « "}{" « height() « "}" «endl;

        while (!tmp.empty())
          {
            os() « "%%\\put(" « tmp.front().xpos()
                 « "," «tmp.front().ypos()
                 «"){\\IPEtext{\\IPEfs{10}\\rm "« tmp.front().text()
                 «" }}" « endl;
            tmp.pop_front();
          }
        os() « "%%\\end{picture}\\endinput}" «endl;
```

```
    }
  os() « "showpage\nend" « endl ;
  cout « flush;
}
```

This macro is defined in definitions 22, 23, 24, and 25.
This macro is invoked in definition 40.

2.4 Implementation : Modifying the State of the Stream

Manipulators are functions inserted into Postscript Stream to change the state of the stream.

The user will write streams like that : `ps « set_line_width(4) « segment;`

The construction of manipulators is :

Manipulators Constructors[26] + ≡

```
#ifndef CGAL_PS_MANIP_DEF
#define CGAL_PS_MANIP_DEF

CGAL_PS_Modifier_creator<CGAL_Color &>
  set_border_color(&CGAL_PS_Stream::_SetBorderColor);

CGAL_PS_Modifier_creator<CGAL_Color &>
  set_fill_color(&CGAL_PS_Stream::_SetFillColor);

CGAL_PS_Modifier_creator<unsigned int>
  set_point_size(&CGAL_PS_Stream::_SetPointSize);

CGAL_PS_Modifier_creator<CGAL_PS_Stream::DotStyle>
  set_point_style(&CGAL_PS_Stream::_SetPointStyle);

CGAL_PS_Modifier_creator<DashStyle>
  set_line_style(&CGAL_PS_Stream::_SetLineStyle);

CGAL_PS_Modifier_creator<unsigned int>
  set_line_width(&CGAL_PS_Stream::_SetLineWidth);

CGAL_PS_Modifier_creator<bool>
  set_fill(&CGAL_PS_Stream::_SetFill);

CGAL_PS_Modifier_creator<const CGAL_PS_Stream::Context &>
  set_current_context(&CGAL_PS_Stream::_SetCurrentContext);

CGAL_PS_Modifier_creator<bool>
  show_direction(&CGAL_PS_Stream::_ShowDirection);

CGAL_PS_Modifier_creator<CGAL_Point_2< CGAL_Cartesian <double> > >
  move_to(&CGAL_PS_Stream::_MoveTo);

CGAL_PS_Modifier_creator<CGAL_PS_Stream::Axis &>
  show_axis(&CGAL_PS_Stream::_ShowAxis);

CGAL_PS_Modifier_creator<CGAL_PS_Stream::Grid &>
```

```
  show_grid(&CGAL_PS_Stream::_ShowGrid);

CGAL_PS_Modifier_creator<const char *>
  put_ps_label(&CGAL_PS_Stream::_PutPsLabel);

CGAL_PS_Modifier_creator<const char *>
  put_latex_label(&CGAL_PS_Stream::_PutLatexLabel);

CGAL_PS_Modifier_creator<unsigned int>
  put_border(&CGAL_PS_Stream::_PutBorder);

CGAL_PS_Modifier_creator<const char *>
  set_font(&CGAL_PS_Stream::_SetFont);

CGAL_PS_Modifier_creator<unsigned int>
  set_font_size(&CGAL_PS_Stream::_SetFontSize);
#endif  //CGAL_PS_MANIP_DEF
```

This macro is defined in definitions 26.
This macro is invoked in definition 40.

This is the class that allows to use manipulators in streams. It is define with a pointer on a function with only one parameter and returning the same type of value. This class contains a field named param that is the value of the modifier inserted in the stream.

We will see further the definitions off the functions called by this way.

Manipulators[27] $+\equiv$

```
#ifndef CGAL_PS_MANIP_H_
#define CGAL_PS_MANIP_H_
class CGAL_PS_Stream;

template <class T>
class CGAL_PS_Modifier {
  friend CGAL_PS_Stream& operator«(CGAL_PS_Stream& pss,
                                   const CGAL_PS_Modifier<T> &);
public:
  CGAL_PS_Modifier(CGAL_PS_Stream& (CGAL_PS_Stream::*f)(T),T v):
    _PS_func(f), param(v) {}
private:
  CGAL_PS_Stream& (CGAL_PS_Stream::*_PS_func)(T);
  T param;
};

template <class T>
CGAL_PS_Stream& operator«(CGAL_PS_Stream& pss, const CGAL_PS_Modifier<T> &m)
{
  (pss.*m._PS_func)(m.param);
  return pss;
};

template<class T>
class CGAL_PS_Modifier_creator {
public:
  CGAL_PS_Modifier_creator(CGAL_PS_Stream& (CGAL_PS_Stream::*f)(T)):
    _PS_func(f) {}
```

```
  CGAL_PS_Modifier<T> operator() (T param)
  {
    return CGAL_PS_Modifier<T>(_PS_func,param);
  }
private:
  CGAL_PS_Stream& (CGAL_PS_Stream::*_PS_func)(T);
};

#endif //CGAL_PS_MANIP_H_
```

This macro is defined in definitions 27.
This macro is invoked in definition 41.

2.5 Implementation : The Catalogue

This function set the default context in the postscript stream.

Non-Template Output Operators[28] + ≡

```
void CGAL_PS_Stream::setdefault()
{
  if (ctxt.get_border_color()!=CGAL_CTXT_DEFAULT.get_border_color())
    os()«"0 0 0 setrgbcolor"«endl;
  if (ctxt.get_line_style()!=CGAL_CTXT_DEFAULT.get_line_style())
    os() « CGAL_PS_Stream::SOLID « " setdash"«endl;
  if (ctxt.get_thickness()!=CGAL_CTXT_DEFAULT.get_thickness())
    os() « 0 « " setlinewidth"«endl;
  if (ctxt.get_font_size()!=CGAL_CTXT_DEFAULT.get_font_size() ||
      strcmp(ctxt.get_font(),CGAL_CTXT_DEFAULT.get_font())!=0)
    {
      os() « "/Helvetica findfont" «endl;
      os() « "12 scalefont setfont" « endl;
    }
  ctxt=CGAL_CTXT_DEFAULT;
}

bool CGAL_PS_Stream::is_eps()
{
  return (bool)(mode()==QUIET_EPS || mode()==READABLE_EPS);
}

bool CGAL_PS_Stream::is_readable()
{
  return (bool)(mode()==READABLE || mode()==READABLE_EPS);
}
```

This macro is defined in definitions 28, 29, and 31.
This macro is invoked in definition 40.

These function insert the postscript catalogue in the file. Thanks to this catalogue the size of the file is reduced.

Non-Template Output Operators[29] + ≡

```
void CGAL_PS_Stream::insert_catalogue()
{
  if (is_eps())
```

```
    {
      os() « "%!PS-Adobe-3.0 EPSF 3.0" « endl;
      os() « "%%BoundingBox: " « "0 0 "
           « width() « " " « height()« endl;
      os() « "%%Creator: CGAL_PS_Stream" « endl;
      os() « "%%Title: (CGAL Output)" « endl;
      os() « "%%CreationDate:" « endl;
    }
  else
    {
      os() « "%!PS-Adobe-3.0" « endl;
    }
  os() « "%%EndComments" «endl«endl;

  // The next line is used to include Latex commands in the file.
  // Thanks to this, it is possible to insert labels in the latex style.
  os() « "{\\catcode37=9\\def\\IPEdummy{({{)}} pop" «endl;
  os() « "%% Ipe postscript prologue" « endl«endl;

  os() « "/CGAL_PS_Dict 14 dict def" « endl;
  os() « "CGAL_PS_Dict begin" « endl;
  os() « "/lt {lineto} bind def" « endl;
  os() « "/mt {moveto} bind def" « endl;
  os() « "/st {stroke} bind def" « endl;
  os() « "/slw {setlinewidth} bind def" « endl;
  os() « "/box {/siz exch def /yy "
       « "exch def /xx exch def xx siz sub yy siz sub siz 2 mul dup} bind def"
« endl;
  os() « "/fb {box rectfill} bind def" « endl;
  os() « "/eb {gsave box 4 copy gsave 1 setgray rectfill grestore "
       « "[] 0 setdash 0 setlinewidth rectstroke grestore} bind def" « endl;
  os() « "/xc {gsave [] 0 setdash 0 setlinewidth "
       « "/siz exch def /yy exch def /xx exch def "
       « "xx siz sub yy siz sub mt xx siz add yy siz add lineto stroke "
       « "xx siz sub yy siz add mt xx siz add yy siz sub lineto "
       « "stroke grestore} bind def" « endl;
  os() « "/ic {gsave [] 0 setdash 0 setlinewidth "
       « "/siz exch def /yy exch def /xx exch def "
       « "xx siz sub yy mt xx siz add yy lineto stroke "
       « "xx yy siz add mt xx yy siz sub "
       «"lineto stroke grestore} bind def" « endl;
  os() « "/cir {0 360 arc} bind def" « endl;
  os() « "/ec {gsave 3 copy gsave 1 setgray cir fill grestore "
       « "[] 0 setdash 0 setlinewidth cir stroke grestore} bind def" « endl;
  os() « "/fc {cir fill} bind def" « endl;
  os() « "/sc {setrgbcolor} bind def" « endl;
  os() « "/tr {mt lt lt lt} bind def" « endl;
  os() « "/re {mt lt lt lt lt} bind def" « endl;
  os() « "0 0 0 setrgbcolor"«endl;
  os() « CGAL_PS_Stream::SOLID « " setdash"«endl;
  os() « 0 « " setlinewidth"«endl;
  os() « "/Helvetica findfont" «endl;
  os() « "12 scalefont setfont" « endl;
```

```
  os() << "0 0 " << width() << " " << height() << " rectclip" << endl;

  setdefault();
}
```

This macro is defined in definitions 28, 29, and 31.
This macro is invoked in definition 40.

2.6 Implementation : Manipulators

Manipulators allow to influence the state of the stream. They permit to change the color, the line width, the line style, the size and the style of the dot when we draw a point, to instert label in the stream, to set the font of the text and its size.

From an implementation point of view there are different solutions. We use a class defining pointers on member fonctions. This is the definition of member function called when a modifier is inserted in the stream.

Non-Template Output Modifiers[30] + ≡

```
CGAL_PS_Stream& CGAL_PS_Stream::_SetBorderColor(CGAL_Color &color)
{
  if (ctxt.get_border_color()!=color)
    {
      os() << color.r() << " " << color.g() << " " << color.b()
           << " setrgbcolor" <<endl;
      ctxt.set_border_color(color);
    }
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetFillColor(CGAL_Color &color)
{
  ctxt.set_fill_color(color);
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetPointSize(unsigned int Size)
{
  ctxt.set_dot_size(Size);
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetLineWidth(unsigned int Width)
{
  if (ctxt.get_thickness()!=Width)
    {
      os()<< Width <<" setlinewidth" <<endl;
      ctxt.set_thickness(Width);
    }
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetPointStyle(DotStyle Style)
{
```

```
  ctxt.set_dot_style(Style);
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetLineStyle(DashStyle style)
{
  if (strcmp(ctxt.get_line_style(),style))
    {
      ctxt.set_line_style(style);
      os() « style « " setdash" « endl;
    }
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetFill(bool test)
{
  ctxt.set_fill(test);
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetDefaultContext(void)
{
  setdefault();
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetCurrentContext(const
CGAL_PS_Stream::Context &c)
{
  if (ctxt.get_border_color()!=c.get_border_color())
    os()« c.get_border_color().r() « " "
        « c.get_border_color().g() « " "
        « c.get_border_color().b() « " setrgbcolor"«endl;
  if (strcmp(ctxt.get_line_style(),c.get_line_style()))
    os() « c.get_line_style() « " setdash"«endl;
  if (ctxt.get_thickness()!=c.get_thickness())
    os() « c.get_thickness() « " setlinewidth"«endl;
  if (ctxt.get_font_size()!=c.get_font_size() ||
      strcmp(ctxt.get_font(),c.get_font())!=0)
    {
      os() « "/" « c.get_font() « " findfont" «endl;
      os() « c.get_font_size() « " scalefont setfont" « endl;
    }
  ctxt=c;
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_ShowDirection(bool choice)
{
  if (choice){};
  return *this;
}
```

```
    CGAL_PS_Stream& CGAL_PS_Stream::_MoveTo(CGAL_Point_2< CGAL_Cartesian <double>
> p)
    {
      ctxt.set_current_pos(p);
      return *this;
    }
    CGAL_PS_Stream& CGAL_PS_Stream::_ShowAxis(Axis &g)
    {
      static bool test=false;
      double x0=x2ps(0);
      double y0=y2ps(0);
      double i;
      if (!test)
        {
          os() « "gsave 0 setgray " « g.thickness()
               « " setlinewidth" « endl;
          os() « "[] 0 setdash" « endl;
          os() « x0 « " " « 0 « " mt" «endl;
          os() « x0 « " " « height() « " lt st" « endl;
          os() « 0 « " " « y0 « " mt" «endl;
          os() « width() « " " « y0 « " lt st" «endl;
          if (g.stepx())
            {
              for (i=((int) (bbox().xmin() / g.stepx())) *g.stepx();
                   i<= bbox().xmax();
                   i+=g.stepx()){
                double x=x2ps(i);
                os() « x « " " « y0 « " mt" «endl;
                os() « x « " " « y0+2 « " lt st" « endl;
              }
            }
          if (g.stepy())
            {
              for (i=((int) (bbox().ymin() / g.stepy())) *g.stepy();
                   i<=bbox().ymax();
                   i+=g.stepy()){
                double y=y2ps(i);
                os() « x0 « " " « y « " mt" «endl;
                os() « x0+2 « " " « y « " lt st" « endl;
              }
            }
          os() « "grestore" «endl;
        }
      test=true;
      return *this;
    }

    CGAL_PS_Stream& CGAL_PS_Stream::_ShowGrid(Grid &g)
    {
      double i;
      os() « "gsave 0 setgray 0 setlinewidth" « endl;
      os() « g.style() « " setdash" « endl;
      if (g.stepx())
```

```
    {
      for (i=((int) (bbox().xmin() / g.stepx())) *g.stepx();
           i<=bbox().xmax();
           i+=g.stepx()){
        double x=x2ps(i);
        os() « x « " 0 mt" «endl;
        os() « x « " " « height() « " lt st" « endl;
      }
    }
  if (g.stepy())
    {
      for (i=((int) (bbox().ymin() / g.stepy())) *g.stepy();
           i<=bbox().ymax();
           i+=g.stepy()){
        double y=y2ps(i);
        os() « " 0 " « y « " mt" «endl;
        os() « width() « " " « y « " lt st" « endl;
      }
    }
  os() « "grestore" «endl;
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_PutPsLabel(const char *ch)
{
  os() « x2ps(context().get_pos().x()) « " "
       « y2ps(context().get_pos().y()) « " mt" «endl;

  os() « "(" « ch « ") show" « endl;
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_PutLatexLabel(const char *ch)
{
  //os() « "%% CGAL - LATEX : " « x2ps(context().get_pos().x()) « " "
  //   « y2ps(context().get_pos().y()) « " " « ch « endl;

  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_PutBorder(unsigned int i)
{
  os() « "gsave" « endl;
  os() « "0 setgray [] 0 setdash" « endl;
  os() « i « " setlinewidth" « endl;
  os() « "0 0 " « width() « " " « height() « " rectstroke" « endl;
  os() « "grestore" « endl;
  return *this;
}

CGAL_PS_Stream& CGAL_PS_Stream::_SetFont(const char *ch)
{
  if (strcmp(ch,context().get_font())!=0)
```

```
      {
        os() « "/" « ch « " findfont" « endl;
        os() « context().get_font_size() « " scalefont setfont" « endl;
        ctxt.set_font(ch);
      }
    return *this;
  }
  CGAL_PS_Stream& CGAL_PS_Stream::_SetFontSize(unsigned int i)
  {
    if (context().get_font_size()!=i)
      {
        ctxt.set_font_size(i);
        os() « "/" « context().get_font() « " findfont" « endl;
        os() « i « " scalefont setfont" «endl;
      }
    return *this;
  }
```

This macro is defined in definitions 30.
This macro is invoked in definition 40.

2.7 Implementation : Output Operators

Each geometric class of CGAL gets its own ouptput operator. We encapsulate each definition in a `#ifdef/#endif` pair so that a user does not implicitely include all geometric header files by including the Postscript stream.

Non-Template Output Operators[31] + ≡

```
  #ifndef _PS_LABEL_
  #define _PS_LABEL_

  CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Border &b)
  {
    ps.os() « "gsave" « endl;
    ps.os() « "0 setgray [] 0 setdash" « endl;
    ps.os() « b.size() « " setlinewidth" « endl;
    ps.os() « "0 0 " « ps.width() « " " « ps.height() « " rectstroke" « endl;
    ps.os() « "grestore" « endl;
    return ps;
  }

  CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Label
&txt)
  {
    ps.os() « ps.x2ps(ps.context().get_pos().x()) « " "
         « ps.y2ps(ps.context().get_pos().y()) « " mt" «endl;

    ps.os() « "(" « txt.text() « ") show" « endl;
    return ps;
  }
```

```
    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, CGAL_PS_Stream::Latex_Label
&txt)
    {
      txt.setposition(ps.x2ps(ps.context().get_pos().x()),
                      ps.y2ps(ps.context().get_pos().y()));
      txt.text();
      ps.list().push_front(txt);
      return ps;
    }

    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Grid &g)
    {
      double i;
      ps.os() « "gsave 0 setgray 0 setlinewidth" « endl;
      ps.os() « g.style() « " setdash" « endl;
      if (g.stepx())
        {
          for (i=((int) (ps.bbox().xmin() / g.stepx())) *g.stepx();
               i<=ps.bbox().xmax();
               i+=g.stepx()){
            double x=ps.x2ps(i);
            ps.os() « x « " 0 mt" «endl;
            ps.os() « x « " " « ps.height() « " lt st" « endl;
          }
        }
      if (g.stepy())
        {
          for (i=((int) (ps.bbox().ymin() / g.stepy())) *g.stepy();
               i<=ps.bbox().ymax();
               i+=g.stepy()){
            double y=ps.y2ps(i);
            ps.os() « " 0 " « y « " mt" «endl;
            ps.os() « ps.width() « " " « y « " lt st" « endl;
          }
        }
      ps.os() « "grestore" «endl;
      return ps;
    }

    CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_PS_Stream::Axis &g)
    {
      static bool test=false;
      double x0=ps.x2ps(0);
      double y0=ps.y2ps(0);
      double i;
      if (!test)
        {
          ps.os() « "gsave 0 setgray " « g.thickness()
               « " setlinewidth" « endl;
          ps.os() « "[] 0 setdash" « endl;
          ps.os() « x0 « " " « 0 « " mt" «endl;
          ps.os() « x0 « " " « ps.height() « " lt st" « endl;
          ps.os() « 0 « " " « y0 « " mt" «endl;
```

```
      ps.os() « ps.width() « " " « y0 « " lt st" «endl;
      if (g.stepx())
        {
          for (i=((int) (ps.bbox().xmin() / g.stepx())) *g.stepx();
               i<= ps.bbox().xmax();
               i+=g.stepx()){
            double x=ps.x2ps(i);
            ps.os() « x « " " « y0 « " mt" «endl;
            ps.os() « x « " " « y0+2 « " lt st" « endl;
          }
        }
      if (g.stepy())
        {
          for (i=((int) (ps.bbox().ymin() / g.stepy())) *g.stepy();
               i<=ps.bbox().ymax();
               i+=g.stepy()){
            double y=ps.y2ps(i);
            ps.os() « x0 « " " « y « " mt" «endl;
            ps.os() « x0+2 « " " « y « " lt st" « endl;
          }
        }
      ps.os() « "grestore" «endl;
    }
  test=true;
  return ps;
}

#endif
```

This macro is defined in definitions 28, 29, and 31.
This macro is invoked in definition 40.

Output Operators[32] + ≡

```
#ifdef CGAL_POINT_2_H

template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_Point_2<R> &p)
{

  if (ps.is_readable())
    {
      ps.os() «  "%CGAL% Point" « endl;
      ps.os() « "%CGAL% "«p.x()«" "«p.y()«endl;
    }
  if (ps.context().get_dot_style()!=CGAL_PS_Stream::NONE)
    ps.os() « ps.x2ps(p.x()) « " "
            « ps.y2ps(p.y()) « " "
            « ps.context().get_dot_size() « " ";
  switch (ps.context().get_dot_style())
    {
    case CGAL_PS_Stream::EBOX:
      ps.os() « "eb" « endl;
      break;
    case CGAL_PS_Stream::FBOX:
```

```
        ps.os() « "fb" « endl;
        break;
      case CGAL_PS_Stream::EDOT:
        ps.os() « "ec" « endl;
        break;
      case CGAL_PS_Stream::FDOT:
        ps.os() « "fc" « endl;
        break;
      case CGAL_PS_Stream::ICROSS:
        ps.os() « "ic" « endl;
        break;
      case CGAL_PS_Stream::XCROSS:
        ps.os() « "xc" « endl;
        break;
      }
  return ps;
}

#endif // CGAL_POINT_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[33] + ≡

```
#ifdef CGAL_SEGMENT_2_H

template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_Segment_2<R> &s)
{
  if (ps.is_readable())
    {
      ps.os() « "%CGAL% Segment" « endl;
      ps.os() « "%CGAL% "«s.source().x()«" "«s.source().y()«" "
              «s.target().x()«" "«s.target().y()«endl;
    }
  ps.os() « ps.x2ps(s.source().x()) « " "
          « ps.y2ps(s.source().y()) « " mt ";
  ps.os() « ps.x2ps(s.target().x()) « " "
          « ps.y2ps(s.target().y())
          « " lt st" « endl;

  return ps;
}

#endif // CGAL_SEGMENT_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[34] + ≡

```
#ifdef CGAL_LINE_2_H

template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_Line_2<R> &l)
{
  if (ps.is_readable())
```

```
    {
      ps.os() « "%CGAL% Line" « endl;
      ps.os() « "%CGAL% "«l.a()«" "«l.b()«" "«l.c()«endl;
    }
  if (!l.is_degenerate())
    if (l.is_vertical())
      {
        double t=ps.x2ps(l.x_at_y(0));
        ps.os()« t;
        ps.os()« " 0 mt" « endl;
        ps.os()« t « " " « ps.height() « " lt st" «endl;
      }
    else
      {
        ps.os() « "0 "
                « ps.y2ps(l.y_at_x(ps.bbox().xmin()))
                « " mt" « endl;
        ps.os() « ps.width() « " "
                « ps.y2ps(l.y_at_x(ps.bbox().xmax()))
                « " lt st"« endl;
      }
  return ps;
}
#endif // CGAL_LINE_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[35] + ≡

```
#ifdef CGAL_RAY_2_H
template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps, const CGAL_Ray_2<R> &r)
{
  typedef CGAL_Direction_2<CGAL_Cartesian <double> > dir;
  CGAL_Line_2<R> l=r.supporting_line();
  dir haut(0,1);
  dir bas(0,-1);
  if (ps.is_readable())
    {
      ps.os() « "%CGAL% Ray" « endl;
      ps.os() « "%CGAL% "«r.source().x()«" "«r.source().y()«" ";
      ps.os() « r.second_point().x() « " " « r.second_point().y() «endl;
    }
  if (!r.is_degenerate())
    {
      ps.os()« ps.x2ps(r.source().x()) « " "
             « ps.y2ps(r.source().y()) « " mt" « endl;
      if (r.is_vertical())
        {
          ps.os()« ps.x2ps(r.source().x()) « " ";
          if (r.direction()==haut)
            ps.os() « ps.height();
          else
            ps.os() « "0 ";
          ps.os() « " lt st" « endl;
```

```
        }
      else
        if (r.direction()>bas || r.direction()<haut)
          ps.os() « ps.width()
                  « " "
                  «ps.y2ps(l.y_at_x(ps.bbox().xmax()))
                  « " lt st" « endl;
        else
          ps.os() « "0 "
                  « ps.y2ps(l.y_at_x(ps.bbox().xmin()))
                  « " lt st" « endl;
    }
  return ps;
}

#endif // CGAL_RAY_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[36] + ≡

```
#ifdef CGAL_PARABOLA_2_H

template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps,const CGAL_Parabola<R> &p)
{

  if (ps.is_readable())
    {
      ps.os() « "%CGAL% Parabola" « endl;
      ps.os() « "%CGAL% Base "«p.base().x()«" "«p.base().y()«endl;
      ps.os() « "%CGAL% Vector "«p.vertor().x()«" "p.vector().y()«endl;
      ps.os() « "%CGAL% Curvature "«p.curvature()«endl;
    }
  return ps;
}X


#endif // CGAL_PARABOLA_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[37] + ≡

```
#ifdef CGAL_TRIANGLE_2_H

template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps,const CGAL_Triangle_2<R> &t)
{
  if (ps.is_readable())
    {
      ps.os() « "%CGAL% Triangle" « endl;
      for (int i=0;i<3;i++)
        ps.os() « "%CGAL " « t[i].x() « " " « t[i].y() « endl;
    }
  for (int i=0;i<4;i++)
```

```
    ps.os() « ps.x2ps(t[i].x())« " " « ps.y2ps(t[i].y()) « " ";

  ps.os() « "tr ";
  if (ps.context().get_fill())
    {
      ps.os() « "gsave " « endl;
      ps.os() « ps.context().get_fill_color().r() « " "
              « ps.context().get_fill_color().g() « " "
              « ps.context().get_fill_color().b()
              « " setcolor fill grestore " «endl;
    }
  ps.os() « "st" «endl;
  return ps;
}


#endif // CGAL_TRIANGLE_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[38] + ≡

```
#ifdef CGAL_ISO_RECTANGLE_2_H

template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps,const CGAL_Iso_rectangle_2<R> &r)
{
    if (ps.is_readable())
      {
        ps.os() « "%CGAL% Rectangle" « endl;
        for (int i=0;i<4;i++)
          ps.os() « "%CGAL " « r[i].x() « " " « r[i].y() « endl;
      }
    for (int i=0;i<5;i++)
      ps.os() « ps.x2ps(r[i].x()) « " " « ps.y2ps(r[i].y()) « " ";

    ps.os() « "re ";
    if (ps.context().get_fill())
      {
        ps.os() « "gsave " « endl;
        ps.os() « ps.context().get_fill_color().r() « " "
                « ps.context().get_fill_color().g() « " "
                « ps.context().get_fill_color().b()
                « " setcolor fill grestore " «endl;
      }
    ps.os() « "st" «endl;
    return ps;
  }
#endif // CGAL_ISO_RECTANGLE_2_H
```

This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

Output Operators[39] + ≡

```
#ifdef CGAL_CIRCLE_2_H
```

```
template < class R >
CGAL_PS_Stream &operator«(CGAL_PS_Stream &ps,
                                   const CGAL_Circle_2<R> &c)
{
    if (ps.is_readable())
      {
        ps.os() « "%CGAL% Circle" « endl;
        ps.os() « "%CGAL " « c.center().x() « " " « c.center().y() « endl;
        ps.os() « "%CGAL " « c.squared_radius() « endl;
      }
    double ratio=ps.yratio()/ps.xratio();
    ps.os()« "gsave 1 " « ratio « " scale" « endl;
    ps.os()« ps.x2ps(c.center().x()) « " " « ps.y2ps(c.center().y())/ratio
           « " " « c.squared_radius()*ps.xratio()  « " 0 360 arc " « endl;
    if (ps.context().get_fill())
      {
        ps.os() « "gsave " « endl;
        ps.os() « ps.context().get_fill_color().r() « " "
                « ps.context().get_fill_color().g() « " "
                « ps.context().get_fill_color().b()
                « " setcolor fill grestore " «endl;
      }
    ps.os() « "st grestore" «endl;
    return ps;
  }
#endif // CGAL_CIRCLE_2_H
```
This macro is defined in definitions 32, 33, 34, 35, 36, 37, 38, and 39.
This macro is invoked in definition 41.

3 Output Files

3.1 The Source File

../src/Postscript_stream.C[40] ≡

```
#include <CGAL/IO/Postscript_stream.h>
```

Constructors and Destructors[22]

Manipulators Constructors[26]

Non-Template Output Modifiers[30]

Non-Template Output Operators[28]
This macro is attached to an output file.

3.2 The Header File

../include/CGAL/IO/Postscript_stream.h[41] ≡
```
#ifndef CGAL_Postscript_STREAM
#define CGAL_Postscript_STREAM
```

```
// For g++ compiler... //

#include <CGAL/Cartesian.h>
#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <iostream.h>
#include <iomanip.h>
#include <fcntl.h>
#include <fstream.h>
#include <strstream.h>
#include <list.h>

#include <LEDA/basic.h>
#include <CGAL/IO/Color.h>
#include <CGAL/Point_2.h>
#include <CGAL/Direction_2.h>
#include <CGAL/Bbox_2.h>
```

Manipulators[27]
class CGAL_PS_Stream[1]
Output Operators[32]

```
#endif  // CGAL_Postscript_STREAM
```

This macro is attached to an output file.