Laporan UTS Program TTY

IF3260 Grafika Komputer

Disusun oleh:

Nicholas Rianto P.	/ 13516020
M. Sulthan Adhipradana	/ 13516035
Yusuf Rahmat P.	/ 13516062
Abram Perdanaputra	/ 13516083
Christian Jonathan	/ 13516092
Faza Fahleraz	/ 13516095

Program Studi Teknik Informatika

Sekolah Teknik Elektro dan Informatika - Institut Teknologi Bandung

Jl. Ganesha 10, Bandung 40132

2019

I. Penjelasan Program

Program yang dibuat adalah sebuah program yang melakukan *assignment* manual terhadap *pixel-pixel* pada layar dengan memanfaatkan TTY (*Teletype*) pada sistem operasi Linux untuk memungkinkan program berinteraksi dengan sistem secara langsung dengan memberikan data, berupa *map* dari setiap *pixel* yang ada pada layar yang berisi kode *warna* yang di*assign* terhadap setiap *pixel* tersebut. Dengan menggunakan konsep tersebut, program dapat digunakan untuk beberapa fitur yang melakukan perubahan langsung terhadap *pixel*.

Gambar 1. Main Menu Program

II. Fitur Program

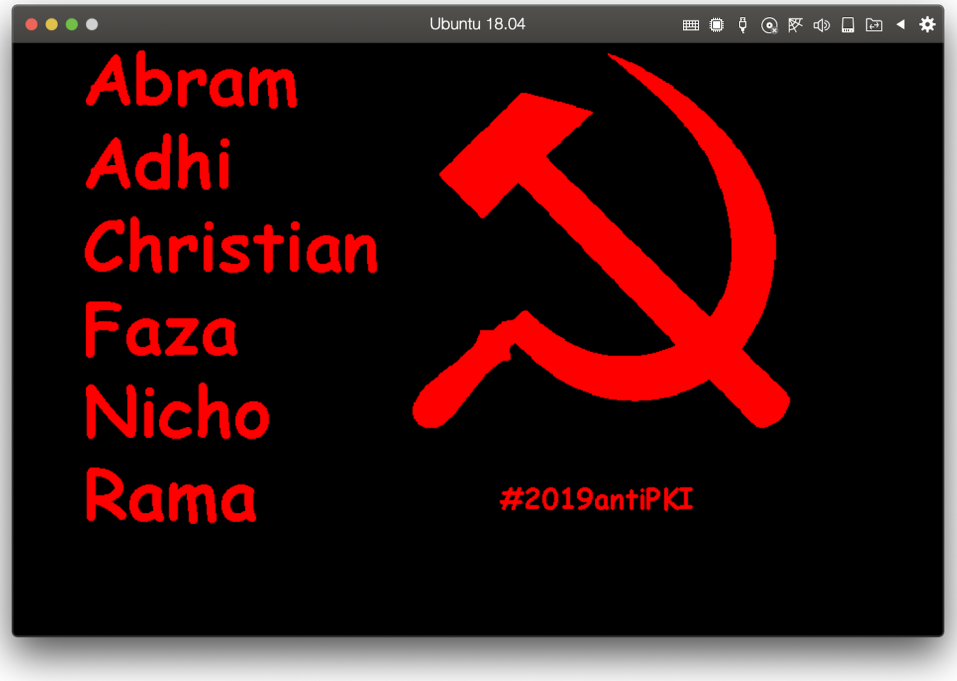
1. Print Point

Program mampu menerima *file* eksternal berupa *list* dari titik koordinat X dan Y (format <*x,y*> setiap *line*nya), dan melakukan *print* point dengan koordinat yang diberikan.

```
110,485
111,485
112,485
113,485
114,485
109,484
110,484
111,484
112,484
113,484
```

Gambar 2. Contoh input point

Program dapat melakukan hal tersebut dengan cara menghitung lokasi pixel yang ditunjuk oleh koordinat (*x , y*). Lokasi dihitung dengan menggunakan data *variable screen information* dan *fixed screen information* dari sistem. Program kemudian melakukan *assignment* berupa kode warna RGB terhadap *pixel* tersebut. Aksi ini dilakukan *looping* sampai semua koordinat yang diterima dari *file* eksternal terproses, dan semua *pixel* pada layar menampilkan warna sesuai *assignment*nya.



Gambar 3. Contoh keluaran Print Point

2. *Print Line*

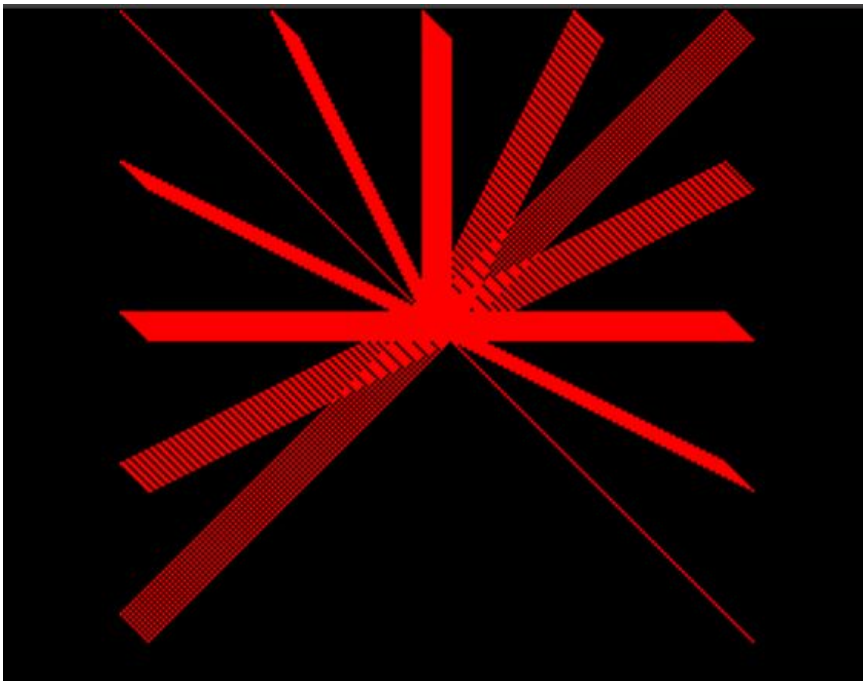
Program mampu menerima *file* eksternal berupa *list* 2 titik koordinat X dan Y yang menunjukkan titik awal dan titik akhir dari sebuah garis (format <*x1*, *y1*, *x2*, *y2*>), dan

melakukan *print* line yang titik awal dan titik akhirnya ditunjuk pada koordinat yang diberikan.

```
120,30,100,70
100,70,80,30
80,30,120,30
130,80,110,50
130,80,150,30
150,30,120,30
90,50,70,80
70,80,40,30
80,30,40,30
90,30,100,20
```

Gambar 4. Contoh input Line

Untuk melakukan *print line*, digunakan fitur pertama yaitu *print point* untuk menggambar seluruh point yang dibutuhkan untuk menghubungkan titik awal dengan titik akhir. Untuk garis lurus horizontal maupun vertikal, program dapat melakukan *print point* untuk point yang sejajar dengan titik awal dan titik akhir. Sedangkan untuk garis diagonal, program mengimplementasi *Bresenham Algorithm* untuk menentukan titik mana yang dilakukan *print point* agar secara keseluruhan membentuk aproksimasi garis yang lurus.

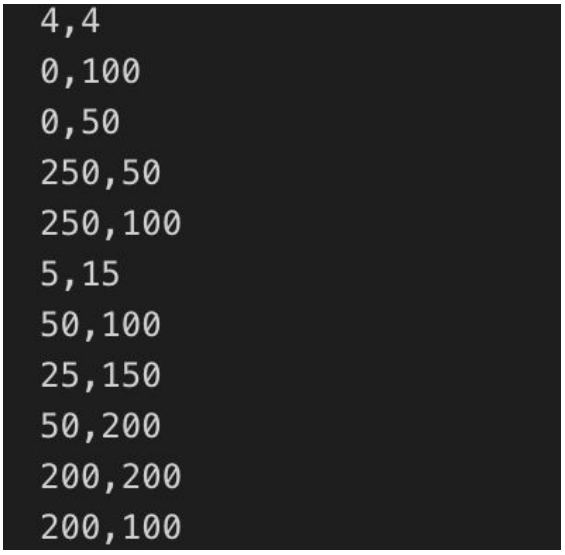


Gambar 5. Contoh keluaran Print Line

Bresenham Algorithm pada dasarnya melakukan iterasi terhadap satu bidang *x*, dan pada setiap iterasinya menghitung apakah lokasi *print point* dilakukan di (*x , y*) atau (*x, y*+1) dengan menghitung apakah gradien garis lebih dekat ke titik (*x , y*) atau (*x, y+1*). Adapun dapat juga dilakukan iterasi terhadap bidang *y* dan menghitung posisi *x* setiap iterasinya. Pemilihan menggunakan bidang yang mana dilakukan dengan menggunakan bidang dengan perubahan yang lebih sedikit sebagai iteratornya.

3. *Print Polygon*

Program menerima *file* eksternal berupa jumlah sisi *polygon*, N, dan menerima sejumlah N garis dimana setiap garisnya berisi koordinat X dan Y setiap titik sudut *polygon* (format <*x,y*>). Kemudian dilakukan *print* polygon tersebut dari titik-titik sudut yang diberikan.



Gambar 6. Contoh input Polygon dengan tambahan Color pada atribut

Untuk melakukan *print polygon*, program menggunakan fitur *print line* untuk melakukan penggambaran garis dari setiap titik sudut ke titik sudut setelahnya, ditambah dari titik sudut terakhir ke titik sudut awal, sehingga terbentuk garis-garis yang menghubungkan setiap titik sudut tersebut membentuk *polygon* yang diinginkan.

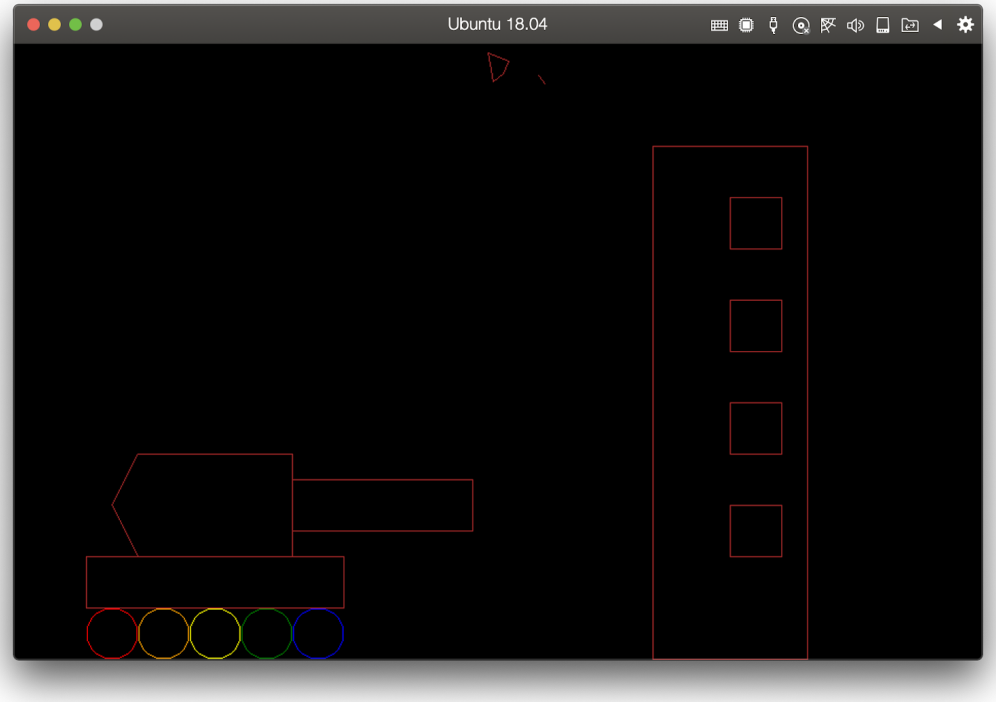
4. *Print Circle*

Program menerima *file* eksternal yang merupakan koordinat titik pusat lingkaran, serta besar radius dari lingkaran yang direpresentasikan (format <*x0, y0, r*>). Program kemudian menggunakan informasi tersebut untuk menggambar lingkaran dengan titik pusat (*x0 , y0*) dan radius *r*.

```
25,25,25,4
75,25,25,5
125,25,25,6
175,25,25,7
225,25,25,8
```

Gambar 7. Contoh input Circle dengan tambahan Color pada atribut terakhir

Program melakukan *print circle* dengan mengimplementasikan *Bresenham's Circle Algorithm*, dimana program mengiterasi dari titik (*r , 0*), melakukan *print point* kemudian mengiterasi *y* ke arah positif, dan pada setiap saat juga mengiterasikan *x* ke arah negatif, sesuai dengan pendekatan persamaan lingkaran, sampai koordinat mencapai (*r/2 , r/2*). Pada iterasi tersebut, program juga melakukan *print point* secara bersamaan pada bagian lingkaran di 8 titik sekaligus sehingga tidak perlu mengiterasi seluruh bagian lingkaran.



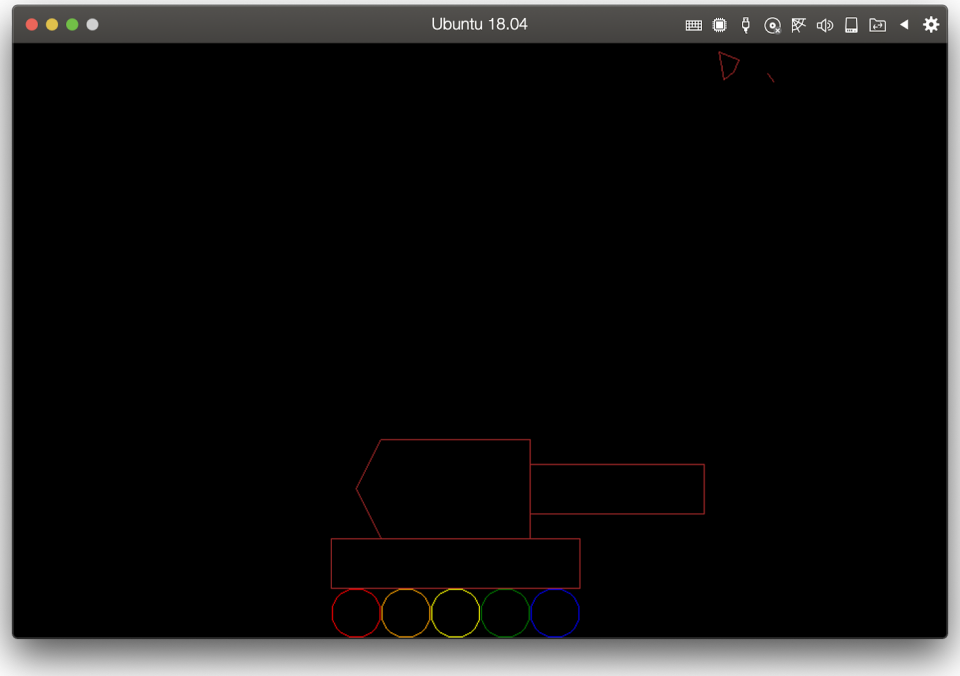
Gambar 8. Contoh keluaran Print Polygon dan Print Circle

5. *Transformation*

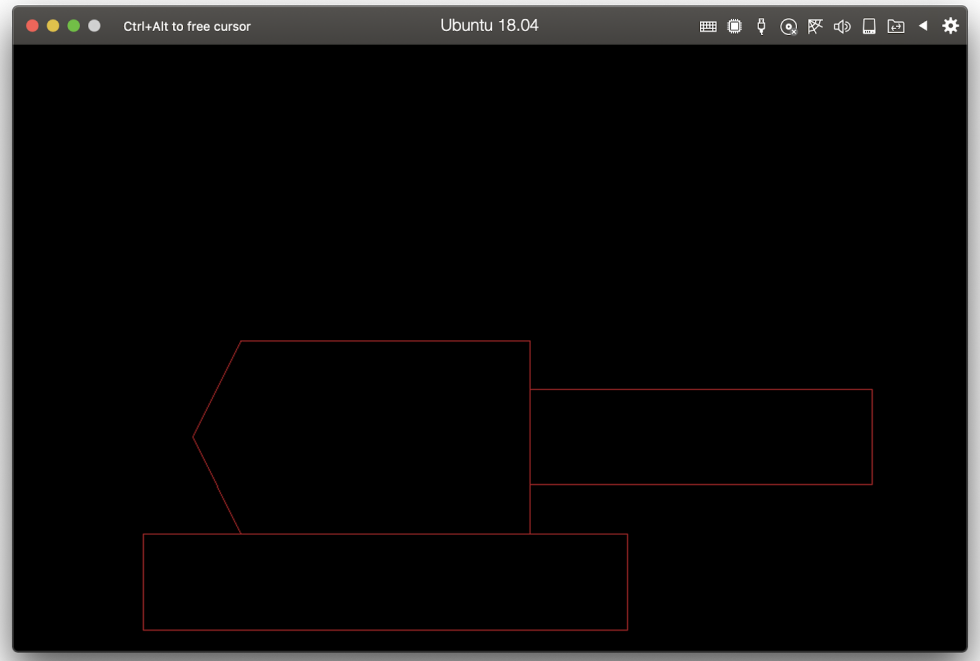
Program mampu menerima *file* eksternal untuk *polygon* dan *circle*, dan setelah menampilkan gambar yang sesuai, dapat melakukan transformasi terhadap seluruh gambar, baik translasi, rotasi (terhadap titik *origin*), dan skalasi (terhadap titik *origin*).

Program dapat melakukan transformasi dengan cara menyimpan seluruh titik yang diterima dari *file* eksternal untuk *polygon* dan *circle*, dan pada setiap perubahan, melakukan *update* nilai untuk semua titik, kemudian menggambarkan ulang semua titik tersebut berdasarkan nilai yang baru.

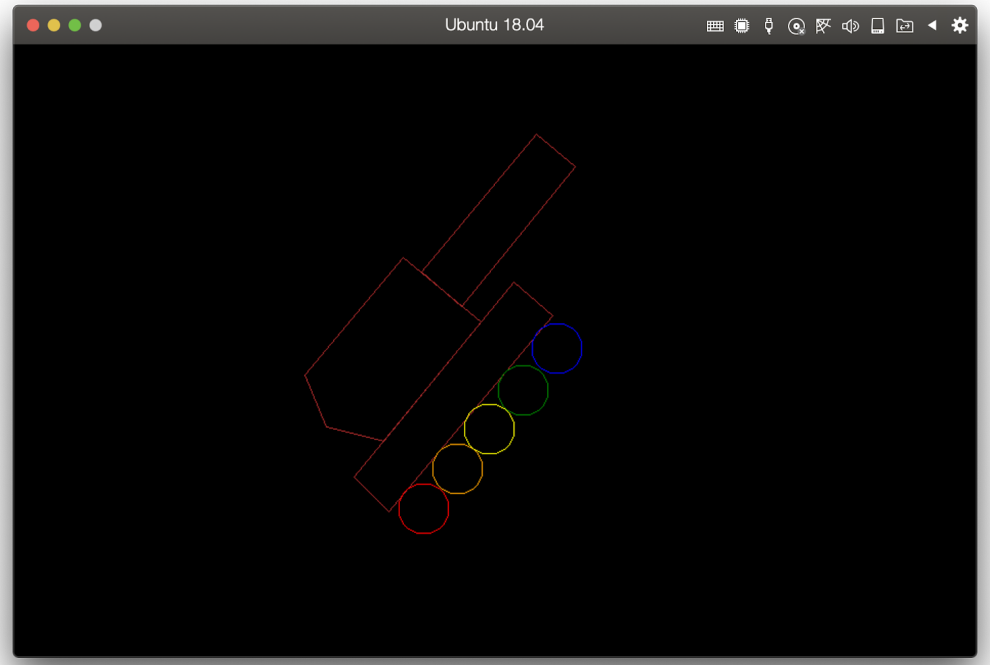
Untuk translasi, pada setiap iterasi, semua koordinat dilakukan penambahan nilai x sebesar dx dan penambahan nilai y sebesar dy. Untuk skalasi, setiap nilai x dan y dilakukan perkalian terhadap k (faktor skala). Untuk rotasi, setiap nilai $x = x * cos(\text{rotasi}) - y * sin(\text{rotasi})$ dan untuk setiap nilai $y = x * sin(\text{rotasi}) + y * cos(\text{rotasi})$.



Gambar 9. Translasi



Gambar 10. Skala



Gambar 11. Rotasi

6. Custom Viewport

Program dapat mengimplementasikan fitur-fitur sebelumnya pada *viewport* dengan ukuran yang telah didefinisikan, dan tidak menampilkan gambar yang berada di luar *viewport*.

Program melakukan *custom viewport* dengan mengimplementasikan *Cohen-Sutherland Clipping Algorithm*, dimana sebelum dilakukan *print line*, setiap segmen garis yang akan diproses dilakukan pengecekkan dengan 3 kemungkinan kondisi, yaitu kondisi pertama seluruh bagian garis berada dalam *viewport*, dimana program akan melakukan *print line* seperti biasa, kondisi kedua seluruh bagian garis berada di luar *viewport*, dimana program akan melakukan *skip* terhadap garis tersebut, dan kondisi ketiga dimana sebagian segmen garis berada di dalam dan sebagian di luar, dimana program akan menghitung titik baru yang menandai segmen garis yang seluruhnya di dalam, dan melakukan *print point* terhadap segmen tersebut.

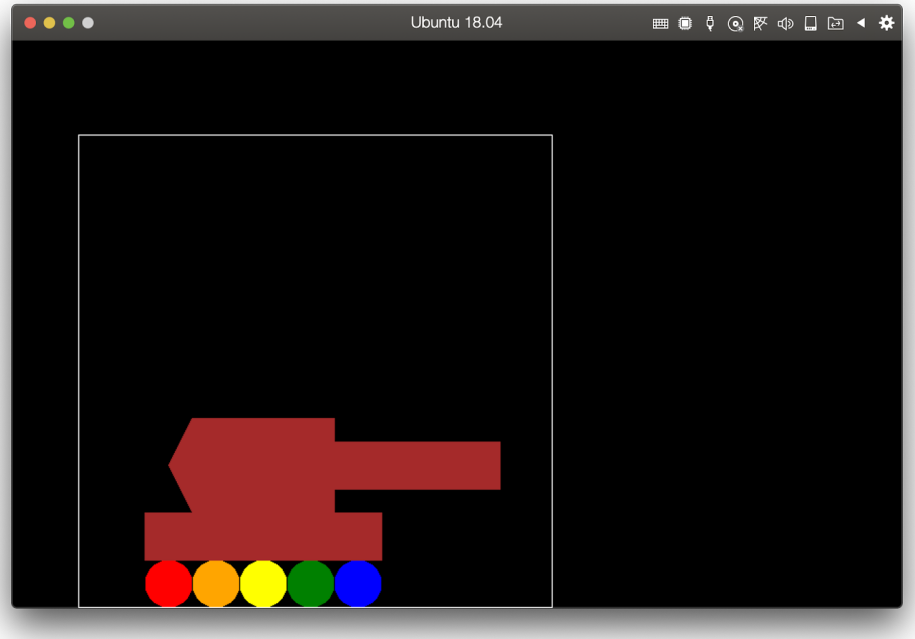
Gambar 12. Implementasi Viewport dengan batasan digambar menggunakan garis putih

7. *Color Fill*

Program dapat menerima *file* eksternal berisi koordinat *polygon* dan *circle*, dengan tambahan atribut warna (berisi ID warna dari *list* warna yang *predefined* pada program), dan program dapat melakukan *filling colors* untuk *polygon* dan *circle* dengan warna yang telah diberikan.

Program melakukan *color fill* dengan mengimplementasikan *Flood Fill Algorithm*. Program akan mengecek setiap bentuk dari *centroid*nya, dan mengiterasi seluruh titik

horizontal sampai *out of bounds* (dibatasi *viewport*) atau memiliki warna yang sama, menandai sudah menunjuk ke batasan bentuknya. Setelah itu, program melakukan pengecekan vertikal pada setiap titik horizontal tersebut, melakukan *assignment* warna kepada setiap *pixel* yang berada di dalam *container* bentuk tersebut.



Gambar 13. Flood Fill pada gambar

III. Pembagian Tugas

NIM	Nama	Tugas
13516020	Nicholas Rianto P.	Membuat print_bmp
13516035	M. Sulthan Adhipradana	Membuat viewport, membuat flood fill, refactor code.
13516062	Yusuf Rahmat P.	Membuat print_polygon, print_circle, transformasi (translasi, skalasi, rotasi). Membuat laporan.
13516083	Abram Perdanaputra	Membuat warna, sistem file eksternal, animasi
13516092	Christian Jonathan	Membuat gambar untuk ditampilkan pada program
13516095	Faza Fahleraz	Melengkapi print_bmp, print_line, flip image, bug fixes, refactors.