Jurnal Mantik
Volume 4, Number 2, Augustus 2020, pp. 1382-1388 E-ISSN 2685-4236
https://iocscience.org/ejournal/index.php/mantik

Image Data Software Component Design and Development for Faster Image Processing

Okky Putra Barus[1], Robin[2]

Teknik Informatika
[1,2]Universitas Pelita Harapan Medan, Lippo Plaza Medan, Lantai 5 - 7, Jl. Imam Bonjol No.6, Petisah Tengah, Medan Petisah, Medan City, North Sumatra 20112

e-mail: robin.huang@lecturer.uph.edu

ARTICLE INFO

Article history:
Received: 12/01/2020
Revised: 22/07/2020
Accepted: 01/08/2020

Keywords:
Image data, faster image, Bitmap, Buffered Image

ABSTRACT

Most software engineers rely too much on operating system (OS) built-in software components when developing softwares. The Bitmap software component is one of those software component classes which is mostly used when developing a software which required to view image which has already provided by the Windows OS. Therefore, using the Bitmap component to process image digitally required longer time to accomplished compares to a very simple image processing software such as Microsoft Paint itself and other image processing tools. This problem has indicates that the Bitmap component isn't suitable and not reliable for the software engineers to develop any softwares which required image processing in their softwares. In this research, this problem can be solved by desining a new image software component which is so called as cImageData which was designed for faster image processing ever. From the experiment result, the cImageData component can process a digital image a lot faster for about 19 times than the Bitmap component do. By using the cImageData software component which can process digital image faster than the built in Bitmap component can provide the engineer to develop a more relyable and higher performance rather than using the Bitmap component in their softwares.

1. Introduction

Digital image processing is one of many essential research field in informatics and computer engineering. Digital image has been widely used since the development of computer graphic technology where digital image processing has been a part of everyday humans live such as multimedia, healthcare, industry, etc.

The Bitmap image format (BMP) has been introduced since the first version of Microsoft Windows at 90s, but the Bitmap software component itself exists since Microsoft .NET Framework 1.1 introduced at 2003 (Microsoft, 2018). The Bitmap component was named Windows Bitmap. The main function of the Bitmap component which was designed by Microsoft is to view image. The Bitmap component also have optional features to gain or change the pixel (picture element) color value. The most important part of image processing is gaining and changing the pixel value while these two optional function in the Bitmap component are not the main function, the Bitmap component is not suitable and not reliable to do a heavy digital image processing because the process of gaining and changing the pixel value is very slow. This matter has caused most image processing and computer vision researcher to use some math applications such as MatLab or other similar tool while those math applications are all interpreters and not compilers. In computer program and software compilation fields, the program which designed using interpreter is an dependent program which required some kind of interpreter software framework or the virtual machine to be installed before the program which is designed by interpreters can be run at computer while the program which is designed using compilers can be compiled into an independent application which does not required any kind of software framework or virtual machine installed on the computers. In this case, because of slower digital imaging proses of the Bitmap component had caused itself not reliable to develop a software which required digital processing in the software.

The Java Development Kit (JDK) also provided the same component as bitmap which is so called BufferedImage. The BufferedImage class has provided almost the same feature as the Bitmap component. It is very different from the Bitmap component where the Bitmap component can be used by any software

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

development tool which run under the Microsoft Windows because the BufferedImage component is only available when developing a Java environment based softwares. The Java based softwares are very rely on the Java Runtime Environment (JRE) framework which is often called Java Virtual Machine (JVM) thus the software application performance become very rely on the JRE. All Java based application cannot be run on computers without the JR. Besides those JRE issues, the similar programs which is designed on different versions of JRE platform also force user has to install the different version of JRE to different OS to make sure all Java based software application can be run smoothly (Oracle, 2018).

Both the Bitmap and BufferedImage component are software components in forms of software unit classes to view image where Bitmap di provided by Microsoft .NET Framework and BufferedImage is provided by JDK. Both class has used a 32 bit integer to store a pixel color value of a digital image where to gain or change each RGB (Red, Green, Blue) color channel value required some arithmetic process and other process to the stored 32 bit integer color value and this process will cost additional unnecessary computer resources and time.

To solve this problem, a fast image processing software component is required to replace the Built in Bitmap component. This new component which is so called cImageData must has an ability process digital image very much faster than the Bitmap without reducing any image data. The cImageData komponen prototype was designed in C++ programming language in compatibility with any C++ programming language in any platform or OS.

2. Previous Research

The BMP image format stored image data in a form of a kind of raster data. The BMP image format which formerly develop for Windows 1.0 does not support color image and does not support any kind of image data compression. This format is designed to support most common IBM-PC graphic card which are popular at the era of CGA, EGA, Hercules graphic cards and monitors. This format is often called device-dependent bitmap (DDB) (Microsoft Docs). The next genertion of Bitmap format is the support for programable 8 bit index color image which enable user define color to be stored as bitmap data. This feature was added since BMP untuk Windows 2.0. When the bitmap image data stored into memory, the information collection is so called Microsoft Device Independent Bimap (DIB) but when they are stored in bitmap image file, it is so called Microsoft Bitmap Format (BMP) (Microsoft, 2018).

The BMP file format can stored 2 dimension monochrome or colored image using a few varitions of color depth such as 1 bit monochrome, 8 bit grayscaled or 8 bit indexed color, 15 bit, 16 bit and 24 bit color depth. It also provided an optional image data compression feature which using the run-length encoding (RLE) lossless method even though most file BMP image file are stored without any compression at all.

Table 1 shows the bitmap image data structure is stored into a BMP image file format. (Murray & VanRyper, 1996)

TABLE 1
THE BMP IMAGE FORMAT

Byte sequence:	function
0 dan 1	Must be 'BM' as the digital signature of BMP image file format
2 to 5	The BMP image filesize
6 to 9	Unused / spared
10 to 13	The beginning byte of data BMP image data stored in BMP image file format.
14 to 17	Unused / spared
18 to 21	BMP image width
22 to 25	BMP image height
26 to 28	Unused / spared
28	Color bit depth of BMP image in bit per pixel metric
29	Unused / spared
30	If the value is 0 means no compression, if value is 1 means the BMP image data stored in BMP image file format is compressed
31 s/d 53	Unused / spared

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Noted that the Bitmap component stored the image data using the Microsoft GDI (Graphics Device Interface) standard and this standard caused programmers can only get each RGB channel color by using features or interfaces which is provided by the GDI such as PixelColo=GetPixel(x, y) and SetPixel(x, y, PixelColor).

Tabel 2 shows essential features which is often used while doing a digital image processing. (Oracle, 2018)

TABLE 2
THE ESSENTIAL FEATURES OF BUFFEREDIMAGE

Feature name	Feature access	Function
Raster Tile	Private Property	To store mage data
int getRGB(x, y);	Public Method	To gain pixel color value
setRGB(x, y, Color);	Public Method	To change pixel color value

Also noted that the BufferedImage component stored the image data in a form of a 32 bit integers rather than storing them in the form array of bytes. This matter caused programmers can only gain each RGB color channel value using some computer arithmetic operations.

3. Methods

The research process was started by gathering images with low resolution until higher image resolution such as 20 megapixel, determining the basic image processing process to test the new cImageData component, determining the essential features of cImageData, designed those feature and tested them through the basic image processing feature determined.

The subject and object in this research are the Bitmap component and BufferedImage component with all its essential feature and also the essential cImageData features which may support advance processing.
The sample digital images to test the performance of each Bitmap, BufferedImage and the new cImageData components is some low resolution images provided by OS and some 20 megapixel digital images which is captured using Digital Camera.

The sample digital images to test the performance of each Bitmap, BufferedImage and the new cImageData components is some low resolution images provided by OS and some 20 megapixel digital images which is captured using Digital Camera.

The data was analyzed using time measurement in OS processor TickCount metric of processing speed on each Bitmap, BufferedImage dan cImageData component using these basic digital image transformation.

1. Flip Horizontal
2. Flip Vertical
3. Rotate 90˚ Clockwise
4. Rotate 180˚ Clockwise
5. Rotate 270˚ Clockwise
6. Zoom 2x

Because basically our computer data are stored in form of bytes, image data access of each RGB color channel can be access faster by using array of bytes. The form of bytes design illustration can be seen in figure 1, figure 2 and figure 3.

As seen in figure 1, figure 2 and figure 3, a pixel at (x,y) coordinates can be accessed using this formula: PixelAddress=(y*width+x)*BytesPerPixel where every cells of those figure represent only 1 byte.

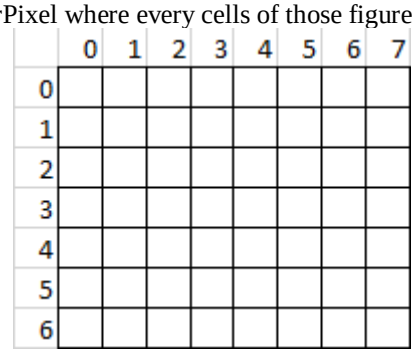

Fig 1. Mapping pixels on a grayscale or 8 bit indexed 7x8 pixel image

Using memory map described in figure 1, an (x,y) pixel can be accessed using this formula: PixelAddress= y*width+x. For example, if there is any grayscale or 8-bit indexed image of 7x8 pixel, according to the formula,

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

the pixel color information on a (6,4) coordinates can be access using this calculation: 4*8+6 to gain the pixel color information at (6,4) coordinates.

	0			1			2		
0	B	G	R	B	G	R	B	G	R
1	B	G	R	B	G	R	B	G	R
2	B	G	R	B	G	R	B	G	R
3	B	G	R	B	G	R	B	G	R
4	B	G	R	B	G	R	B	G	R
5	B	G	R	B	G	R	B	G	R
6	B	G	R	B	G	R	B	G	R

Figure 2. Mapping pixels on a 24 bit 3x6 pixel image

Using memory map described in figure 2, an (x,y) pixel can be accessed using this formula: PixelAddress= (y*width+x)*3 because 24 bit depth colored image required 3 bytes to store a pixel color information. For example, if there is 24-bit image of 3x6 pixel, according to the formula, the pixel color information on a (2,5) coordinates can be access using this calculation: (5*7+2)*3 to gain the pixel color information at (2,5) coordinates.

	0				1			
0	A	B	G	R	A	B	G	R
1	A	B	G	R	A	B	G	R
2	A	B	G	R	A	B	G	R
3	A	B	G	R	A	B	G	R
4	A	B	G	R	A	B	G	R
5	A	B	G	R	A	B	G	R
6	A	B	G	R	A	B	G	R

Figure 3. Mapping pixels on 32 bit 2x6 pixel image

Using memory map described in figure 3, an (x,y) pixel can be accessed using this formula: PixelAddress= (y*width+x)*4 because 32 bit depth colored image required 4 bytes to store a pixel color information. For example, if there is 32-bit image of 2x6 pixel, according to the formula, the pixel color information on a (1,3) coordinates can be access using this calculation: (3*7+1)*4 to gain the pixel color information at (1,3) coordinates.

The cImageData prototype was coded in C++ programming language can be seen below.

```
class cImageData {
 private:  //private property
  unsigned char *Pixels;
  int Address;
  TPixelFormat PixelFormat;

 public:  //public property
  int Width, Height, BitPerPixel, BytePerPixel;

  //constructor
  cImageData();
  cImageData(int InitWidth, int InitHeight, int BPP);
  cImageData(int InitWidth, int InitHeight, int BPP, TPixelFormat tf);

  //common method
  void AllocateMemory(int InitWidth, int InitHeight);
  void AllocateMemory(int Size);
  void AllocateMemory();
  void CopyFrom(cImageData *DataSrc);

  //accessor
  int CalculateAddress(int x, int y);
```


Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

```
  void ChangeDataAddress(int x, int y);
  int GetPixel(int x, int y);
  void GetARGB(int x, int y, unsigned char *A, unsigned char *R, unsigned
char *G, unsigned char *B);
  void GetRGB(int x, int y, unsigned char *R, unsigned char *G, unsigned
char *B);
  unsigned char GetByte(int x, int y);
  unsigned char GetR(int x, int y);
  unsigned char GetG(int x, int y);
  unsigned char GetB(int x, int y);
  unsigned char GetA(int x, int y);
  int GetWidth();
  int GetHeight();
  int GetBytePerPixel();
  int GetBitPerPixel();
  unsigned char GetCurrentByte();
  unsigned char GetCurrentR();
  unsigned char GetCurrentG();
  unsigned char GetCurrentB();
  unsigned char GetCurrentA();

  //mutator
  void SetPixel(int x, int y, unsigned char A, unsigned char R, unsigned
char G, unsigned char B);
  void SetPixel(int x, int y, unsigned char R, unsigned char G, unsigned
char B);
  void SetPixel(int x1, int y1, cImageData *Src, int x2, int y2);
  void SetARGB(int x, int y, unsigned char A, unsigned char R, unsigned
char G, unsigned char B);
  void SetRGB(int x, int y, unsigned char R, unsigned char G, unsigned
char B);
  void SetByte(int x, int y, unsigned char ByteValue);
  void SetR(int x, int y, unsigned char Red);
  void SetG(int x, int y, unsigned char Green);
  void SetB(int x, int y, unsigned char Blue);
  void SetA(int x, int y, unsigned char Alpha);
  void SetCurrentByte(unsigned char ByteValue);
  void SetCurrentR(unsigned char Red);
  void SetCurrentG(unsigned char Green);
  void SetCurrentB(unsigned char Blue);
  void SetCurrentA(unsigned char Alpha);
  bool LoadFromBitmap(unsigned char FileName[]);
  void CopyFromBitmap(Graphics::TBitmap *Bitmap);
  void CopyToBitmap(Graphics::TBitmap *Bitmap);
  void CopyToBitmap(Graphics::TBitmap *Bitmap, TPixelFormat tf);
  bool Convert8BitTo24Bit(cImageData *ImageResult);
  bool Convert24BitTo8Bit (cImageData *ImageResult);
  bool Convert24BitTo32Bit(cImageData *ImageResult);
  bool Convert32BitTo8Bit (cImageData *ImageResult);
  bool Convert32BitTo24Bit(cImageData *ImageResult);
  cImageData::~cImageData();
};
```

4. Results and Discussion

Table 3, Table 4 and Table 5 shows comparison between the three component in OS processor TickCount metric on 640x480, 800x600 and 1024x768 pixel 24 bit RGB colored image and Table 6 shows comparison between them on 20 megapixel (3840x2160 pixel) 24 bit RGB colored image running on these computer hardware specification below.

Intel Core2Duo Processor T6500 2.1Ghz / 800MHz FSB / Dual Core
Intel Express Chipset 965
Mobile Intel ® 4 Series Express Chipset Graphic
DDR2 RAM 3GB
Harddisk Drive 500GB

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

TABLE 3.
AVERAGE PROCESSING TIME BETWEEN THOSE THREE COMPONENT ON 640X480 24 BIT RGB COLORED IMAGE IN TICKCOUNT METRIC

Transformation	Windows Bitmap	Java BufferedImage	cImageData
Flip Horizontal	656	93	31
Flip Vertical	656	93	31
Rotate 90˚ clockwise	656	93	31
Rotate 180˚ clockwise	656	93	31
Rotate 270˚ clockwise	656	93	31
Zoom 2x	3360	265	219

TABLE 4.
AVERAGE PROCESSING TIME BETWEEN THOSE THREE COMPONENT ON 800X600 24 BIT RGB COLORED IMAGE IN TICKCOUNT METRIC

Transformation	Windows Bitmap	BufferedImage Java	cImageData
Flip Horizontal	922	125	47
Flip Vertical	922	125	47
Rotate 90˚ clockwise	922	125	47
Rotate 180˚ clockwise	922	125	47
Rotate 270˚ clockwise	922	125	47
Zoom 2x	4469	390	234

TABLE 5.
AVERAGE PROCESSING TIME BETWEEN THOSE THREE COMPONENT ON 1024X768 24 BIT RGB COLORED IMAGE IN TICKCOUNT METRIC

Transformation	Windows Bitmap	BufferedImage Java	cImageData
Flip Horizontal	1547	172	78
Flip Vertical	1547	235	78
Rotate 90˚	2078	172	93
Rotate 180˚	2078	172	93
Rotate 270˚	2078	172	93
Zoom 2x	4469	578	438

TABLE 6
AVERAGE PROCESSING TIME BETWEEN THOSE THREE COMPONENT ON 20 MEGAPIXEL (3840X2160 PIXEL) 24 BIT RGB COLORED IMAGE IN TICKCOUNT METRIC

Proses Transformasi Dasar	Windows Bitmap	BufferedImage Java	cImageData
Flip Horizontal	21016	2266	328
Flip Vertical	21016	2266	328
Rotate 90˚	21016	2266	516
Rotate 180˚	21016	2266	516
Rotate 270˚	21016	2266	516
Zoom 2x	32657	Not responding	1828

Only by testing through basic geometry transformation process such as flip, rotate and zoom Table 1 to Table 4, shows that the Bitmap component required about 4.6 times (slower and longer time required) than the BufferedImage component and Windows Bitmap required at lease about 19.8 times (slower and much more longer time required) than the processing time required by the cImageData component.

5. Conclusion

After a short discussion, here are a few conclusions gained after this experiment.

1. By analyzing the data gained from this experiment in table 3, table 4, table 5 and table 6, it is proven that the Bitmap component is not suitable and not reliable to be applied any complex image processing algorithm while the BufferedImage and cImageData components are proven to be more suitable and more reliable to be applied any complex image processing algorithm.
2. In software multimedia-based development world, by using the cImageData component can certainly process digital image faster than the Bitmap and BufferedImage component and this is the reason why multimedia-based software can process digital image data faster using the cImageData component.

Accredited "Rank 4"(Sinta 4), DIKTI, No. 36/E/KPT/2019, December 13th 2019.

Acknowdlegment

The authors would like thank the Ministry of Research and Technology / the National Research and Innovation Agency for the funding provided through the Power of Attorney Decree Number 045 / 01 / LPPM-M / VIII / 2020. The authors also would like to thank Ferawaty from Universitas Pelita Harapan, Wenripin from STMIK Mikroskil and Saut Dohot Siregar from Universitas Prima Indonesia for their support and help in advising this article.

6. References

[1] Chapman N., & Chapman J., 2009. Digital Multimedia. In N. Chapman, & J. Chapman, Digital Multimedia (p. 736). Highlands: John Wiley & Sons, Ltd.
[2] GNU Classpath, 2018, BufferedImage. Retrieved from GNU Classpath: http://developer.classpath.org/doc/java/awt/image/BufferedImage-source.html
[3] Microsoft, 2018, Bitmap Drawing. Retrieved from Microsoft Docs: https://msdn.microsoft.com/en-us/library/system.drawing.bitmap(v=vs.110).aspx
[4] Microsoft, 2018, class Bitmap. Retrieved from Microsoft Docs: https://docs.microsoft.com/en-us/windows/win32/api/gdiplusheaders/nl-gdiplusheaders-bitmap?redirectedfrom=MSDN
[5] Microsoft, 2018, NET Framework. Retrieved from NET Framework: https://docs.microsoft.com/en-us/dotnet/framework/migration-guide/versions-and-dependencies
[6] Murray JD, & VanRyper W, 1996, Encyclopedia of Graphics File Format. In J. D. Murray, & W. VanRyper, Encyclopedia of Graphics File Format. O'Reilly & Associates, Inc.
[7] Oracle, 2018, Oracle Docs. Retrieved from class BufferedImage: https://docs.oracle.com/en/java/javase/11/docs/api/java.desktop/java/awt/image/BufferedImage.html
[8] Yuan, Feng, 2000, Windows Graphics Programming Win32 GDI and DirectDraw
[9] Microsoft Developer Microsoft. 2018 Retrieved from https://msdn.microsoft.com/en-us/library/dd183385(v=vs.85).aspx
[10] Kumar SN, Sangaiah AK, Arun M, Anand S, 2017, Advanced Image Processing Techniques and Applications, IGI Global
[11] Furth B, Akar E, Andrews WA, 2018, Digital Image Processing, Practical Approach Springer
[12] Russ JC, Neal FB, 2016, The Image Processing Handbook, CRC Press
[13] Pratt WK, 2014, Introduction to Digital Image Processing, CRC Press
[14] Hidayatullah P, 2017, Pengolahan Citra Digital, Informatika
[15] Robin, Suharjito, 2015, Optimized Weighed Matrix of Non-Negative Matrix Factorization for Image Compression, International Journal of Multimedia and Ubiquitous Engineering
[16] Robin, Ferawaty, Jusin, Hita, Irviantina S, Chandra W, Siregar SD, 2019, An Empiric Model of Face Detection based on RGB Skin Tone Color, IOP Publishing