**ĐẠI HỌC QUỐC GIA THÀNH PHỐ HỒ CHÍ MINH**

**ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**

**KHOA CÔNG NGHỆ PHẦN MỀM**



**BÁO CÁO ĐỒ ÁN CUỐI KỲ**

**MÔN: LẬP TRÌNH TRỰC QUAN**

**ĐỀ TÀI: PHẦN MỀM PAINT**

**GIẢNG VIÊN HƯỚNG DẪN**

* Thầy Lê Thanh Trọng
* Thầy Huỳnh Tuấn Anh

**SINH VIÊN THỰC HIỆN**

Bùi Đình Lộc thọ 13520844

Lê Tấn Thịnh 13520836

**PHỤ LỤC**

[**I – GIỚI THIỆU** 3](#_Toc421797627)

[**1.Giới thiệu phần mềm Microsoft Paint** 3](#_Toc421797628)

[**2.Giới thiệu chủ đề** 3](#_Toc421797629)

[**II – QUY TRÌNH** 3](#_Toc421797630)

[**1.** **Phân chia công việc** 3](#_Toc421797631)

[**2.** **Giới thiệu GDI+** 4](#_Toc421797632)

[**3.** **Thiết kế phần mềm** 4](#_Toc421797633)

[**3.1 Màn hình** 4](#_Toc421797634)

[**3.2 Chức năng Chính** 5](#_Toc421797635)

[**3.3 Chức năng Menu** 6](#_Toc421797636)

[**4.** **Luồng sự kiện** 10](#_Toc421797637)

[**5.** **Mô phỏng chức năng** 11](#_Toc421797638)

[**5.1** **Marquee** 11](#_Toc421797639)

[**5.2** **Fill** 12](#_Toc421797640)

[**5.3** **Eraser** 13](#_Toc421797641)

[**5.4** **Brush** 14](#_Toc421797642)

[**5.5** **Zoom** 15](#_Toc421797643)

[**5.6** **Text** 16](#_Toc421797644)

[**5.7** **Color Picker** 17](#_Toc421797645)

[**5.8** **Shape** 18](#_Toc421797646)

[**5.9 Colors** 20](#_Toc421797647)

[**6.** **Kĩ thuật** 20](#_Toc421797648)

[**6.1** **Thuật toán Flood Fill** 20](#_Toc421797649)

[**6.2** **Lock bits** 21](#_Toc421797650)

[**III – TỔNG KẾT** 21](#_Toc421797651)

[**1.** **Đánh giá** 21](#_Toc421797652)

[**2.** **Website và tài liệu tham khảo** 21](#_Toc421797653)

# **I – GIỚI THIỆU**

## **1.Giới thiệu phần mềm Microsoft Paint**

Paint (trước đây là Paintbrush for Windows) là một chương trình đồ họa máy tính đơn giản, đã được tích hợp sẵn trong tất cả các phiên bản của Microsoft Windows(thường được gọi tắt là MS Paint hay Microsoft Paint). Chương trình này chủ yếu dùng để mở và lưu các tập tin như Windows bitmap (24-bit, 256 color, 16 color, and monochrome, all with the .bmp extension), JPEG, GIF, PNG và single-page TIFF. Chương trình có hai chế độ: ảnh màu hoặc trắng-đen, không có chế độ grayscale. MS Paint nhanh chóng trở thành một trong những ứng dụng được sử dụng nhiều nhất trong các phiên bản đầu tiên của Windows vì tính đơn giản của nó và hiện tại vẫn được sử dụng rộng rãi cho các tác vụ xử lý ảnh đơn giản.

## **2.Giới thiệu chủ đề**

Mục tiêu của đồ án là thiết kế một phần mềm mô phỏng lại một sô chức năng của phần mềm Microsoft Paint như: Vẽ, vẽ đường thẳng, vẽ các hình đa giác, tô màu, mở file, lưu file v.v…

# **II – QUY TRÌNH**

## **Phân chia công việc**

|  |  |
| --- | --- |
| **Họ và tên** | **Công Việc** |
| Bùi Đình Lộc Thọ | - Thiết kế và Code về ToolBox và các Chức năng trên ToolBox.  - Viết báo cáo:  luồng sự kiện  Mô phỏng chức năng  Kĩ thuật  Tổng Kết  - Test, Fix bugs chương trình  - Viết Slides. |
| Lê Tấn Thịnh | Thiết kế và Code về Menu và các chức năng trên Menu.  - Viết báo cáo:  Giới thiệu  Giới Thiệu GDI+  Thiết kế phần mềm  - Test, Fix bugs chương trình  - Viết Slides. |

## **Giới thiệu GDI+**

- GDI (Graphics Development Interface) là các API của Windows cung cấp các hàm và các CTDL cần thiết để tạo ra những hình ảnh đồ họa 1 cách nhanh chóng và hiệu quả ra các thiết bị phần cứng (như màn hình, máy in).

- GDI+ cải thiện GDI với các đặc điểm mới và tối tưu các đặc điểm đã có ở GDI.

- Trong Microsoft Visual Studio, GDI+ có trong khối hợp ngữ System.Drawing.dll, 1 vài namespace trong đó là:

System.Drawing.Design;

System.Drawing;

System.Drawing.Imaging;

System.Drawing.Drawing2D;

System.Drawing.Printing;

System.Drawing.Text;

**- Các thư viện sử dụng trong đồ án:**

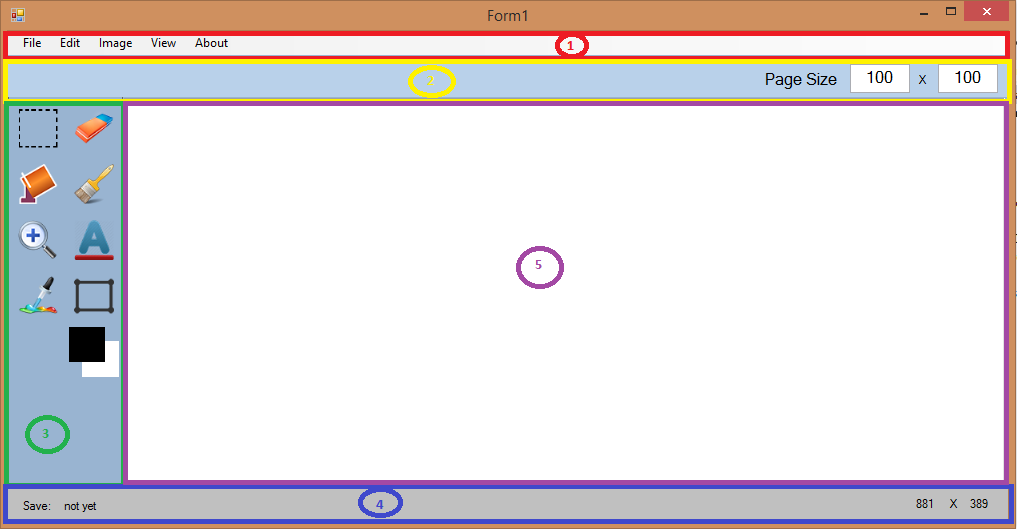
System.Drawing;

System.Drawing.Imaging;

## **Thiết kế phần mềm**

### **3.1 Màn hình**

- Phần mềm có 1 màn hình chính như dưới.



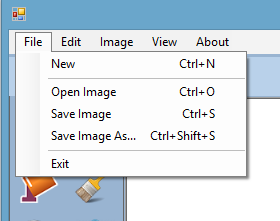
|  |  |  |
| --- | --- | --- |
| **STT** | **Tên** | **Chức năng** |
| 1 | MenuBar | Chứa các menu để người dùng thao tác lên bảng vẽ như(New, Open, Save, Undo, Redo, Convert…) |
| 2 | PropertiesBar | Hiển thị các thuộc tính của công cụ mà người dùng chọn, từ đó có thể chỉnh sửa sao cho phù hợp |
| 3 | ToolBox | Chứa các công cụ hỗ trợ vẽ ( như: Brush, Eraser, Marquee, Zoom, Fill, Text, Shape, LeftColor, RightColor, ColorPicker) |
| 4 | StatusBar | Hiện thị trạng thái của bảng vẽ(Kích thước của trang vẽ, đã được lưu hay chưa) |
| 5 | WorkSpace | Bảng vẽ (Người dùng sử dụng các công cụ để vẽ lên đây) |

### **3.2 Chức năng Chính**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **STT** | **Tên** | **Biểu tượng** | **Loại Control** | **Chức năng** |
| 1 | Marquee | F:\Github\Paint\Paint-Crazyland\Resources\Marquee-Rectangle.png | Button | Hỗ trợ cắt và di chuyển 1 phần ảnh trên WorkSpace |
| 2 | Fill | F:\Github\Paint\Paint-Crazyland\Resources\fill-tool.png | Button | Tô màu trên 1 vùng |
| 3 | Eraser | F:\Github\Paint\Paint-Crazyland\Resources\eraser-tool.png | Button | Cục tẩy, dùng để xóa các nét vẽ |
| 4 | Brush | F:\Github\Paint\Paint-Crazyland\Resources\brush-tool.png | Button | Cọ vẽ, dùng để vẽ lên WorkSpace |
| 5 | Zoom | F:\Github\Paint\Paint-Crazyland\Resources\zoom-tool.png | Button | Hỗ trợ phóng to ảnh trên WorkSpace |
| 6 | Text | F:\Github\Paint\Paint-Crazyland\Resources\text-tool.png | Button | TextBox để nhập các ký tự |
| 7 | Color Picker | F:\Github\Paint\Paint-Crazyland\Resources\colorpicker.png | Button | Lấy màu tại vị trí click chuột và cài đặt màu cho LeftColor |
| 8 | Shape | F:\Github\Paint\Paint-Crazyland\Resources\rectangle_stroked.png | Button | Hỗ vẽ các đường thẳng, hình tam giác, tứ giác, ngũ giác, lục giác lên WorkSpace |
| 9 | Colors |  | Button | Hiển thị màu hiện tại(LeftColor: chuột trái, RightColor: chuột phải) của công cụ vẽ, khi click vào sẽ hiển thị hộp thoại chọn màu |

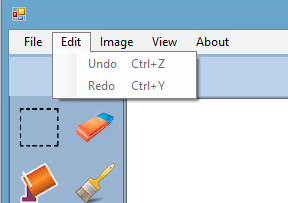
### **3.3 Chức năng Menu**

#### **3.3.1 Menu File**



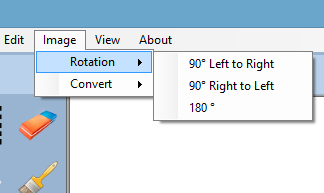
|  |  |
| --- | --- |
| **Tên MenuItem** | **Chức năng** |
| New | Tạo WorkSpace mới |
| Open Image | Mở một hình có sẵn |
| Save Image | Lưu WorkSpace |
| Save Image As | Lưu WorkSpace dưới dạng.... |
| Exit | Thoát chương trình |

#### **3.3.2 Menu Edit**

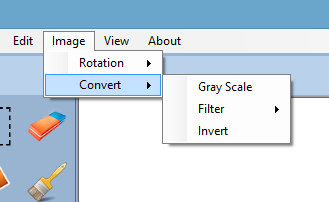


|  |  |
| --- | --- |
| **Tên MenuItem** | **Chức năng** |
| Undo | Lùi về trước 1 thao tác |
| Redo | Tiến trước 1 thao tác |

#### **3.3.3 Menu Image**

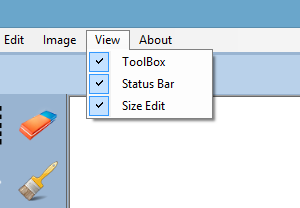
****

|  |  |
| --- | --- |
| **Tên MenuItem** | **Chức năng** |
| 90° Left to Right | Xoay ảnh sang phải 90 độ |
| 90° Right to Left | Xoay ảnh sang trái 90 độ |



|  |  |
| --- | --- |
| **Tên MenuItem** | **Chức năng** |
| Invert | Chuyển sang màu âm bảng |
| Filter | Lọc màu |
| Gray Scale | Chuyển sang ảnh xám |

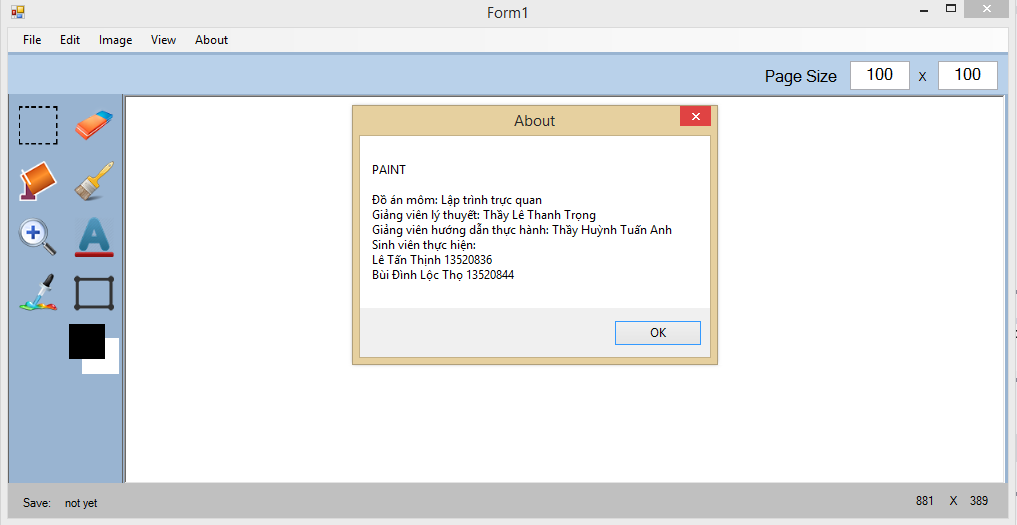
#### **3.3.4 Menu View**

****

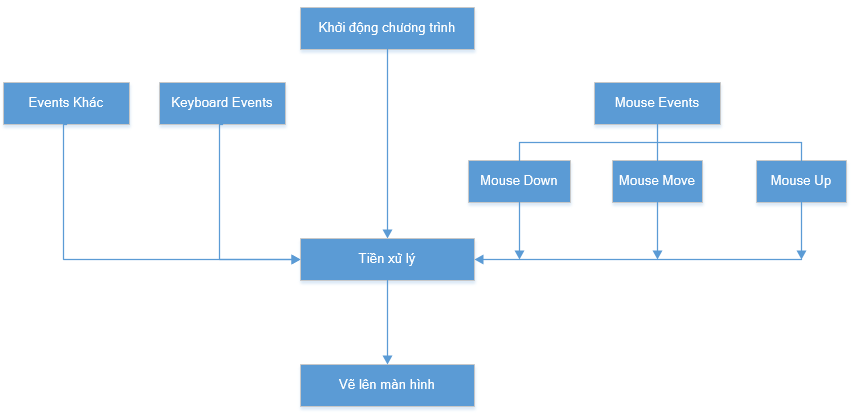
|  |  |
| --- | --- |
| **Tên MenuItem** | **Chức năng** |
| ToolBox | Ẩn, hiện thanh ToolBox |
| StatusBar | Ẩn, hiện thanh StatusBar |
| Size Edit | Ẩn, hiện công cụ thay đổi kích thước WorkSpace (Page Size nằm bên góc phải) |

#### **3.3.5 Menu About**

- Thông tin về tác giả và giảng viên hướng dẫn



## **Luồng sự kiện**



**- Khối khởi động chương trình:** lúc chương trình được mở lên khởi tạo các thành phần cần thiết cho chương trình.

**- Khối Keyboard Events:** các sự kiện từ bàn phím

**- Khối Mouse Events:** các sự kiện từ chuột, chia ra làm 3 event nhỏ

+ Mouse Down: sự kiện khi chuột được nhấp xuống, ghi nhận tọa dộ chuột, và xử lý những thứ cần thiết.

+ Mouse Move: sự kiện khi chuột di chuyển, ghi nhận sự kiện chuột, tùy thuộc vào loại công cụ đang được chọn để xử lý những thứ cần thiết.

+ Mouse Up: sự kiện khi chuột được nhả ra, ghi nhận sự kiện chuột, tùy thuộc vào loại công cụ đang được chọn để xử lý những thứ cần thiết.

**- Khối Events Khác:** các sự kiện khác của các Control như sự kiện Button Clicked, MenuItem Clicked, TrackBar Scroll….

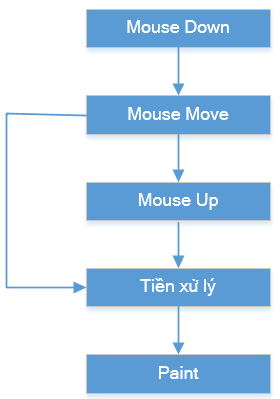
**- Khối tiền xử lý:** xử lý chương trình, các sự kiện trước khi vẽ lên màn hình.

**- Vẽ lên màn hình:** sau khi xử lý các sự kiện xong bắt đầu vẽ lên màn hình, ở đây sẽ là việc thực hiện các hàm, lớp của GDI+, đặc biệt là đôi tượng Graphics.

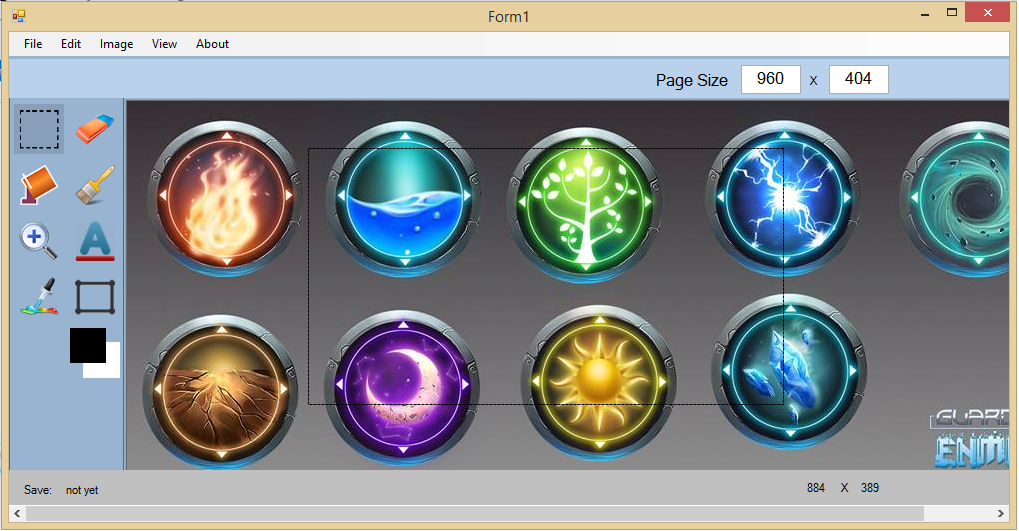
## **Mô phỏng chức năng**

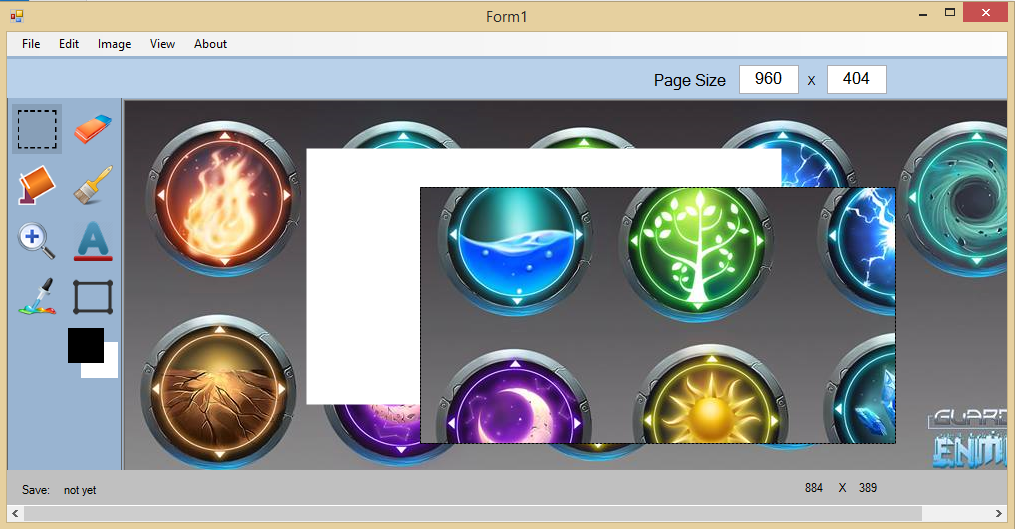
### **Marquee**

- Sơ đồ hoạt động

****

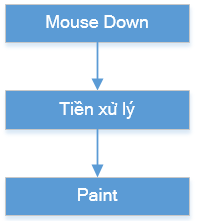
- Đầu tiên sẽ bắt sự kiện Mouse Down lấy tọa độ điểm và lưu lại, tiếp đến di chuyển chuột để khoanh vùng cắt ảnh lúc này thì ta cũng gọi đến hàm Paint để vẽ lên 1 hình chữ nhật với đường viền nét đứt để mô phỏng vùng bị cắt và vùng bị cắt sẽ được xác định khi sự kiện Mouse Up được thực hiện. Sau khi xác định vùng xong ta có thể nhấp giữ vùng chọn để di chuyển đến vị trí mong muốn.



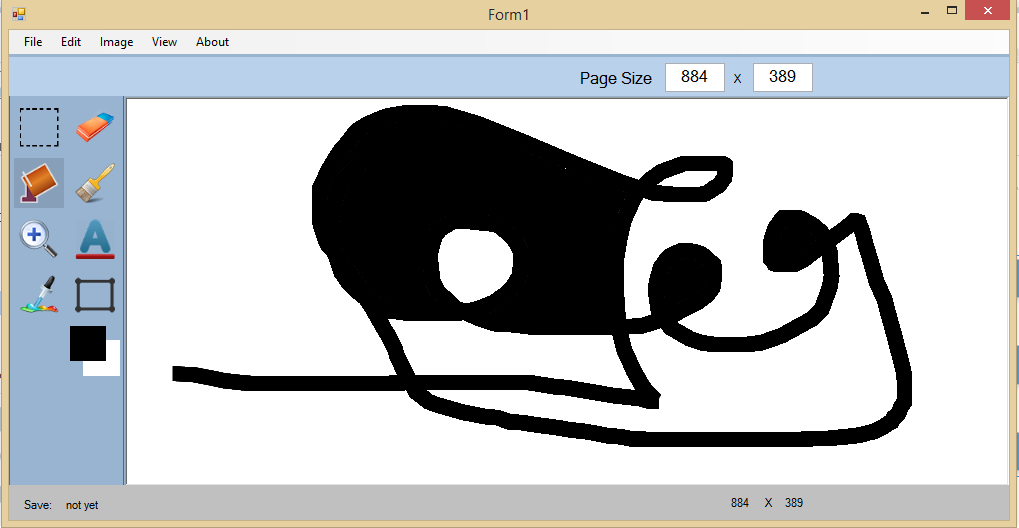


### **Fill**

- Sơ đồ hoạt động

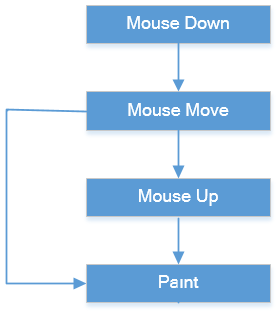


- Tiếp nhận sự kiện MouseDown lấy tọa độ chuột, sau đó xử lý LockBits thực hiện thuật toán tô màu Flood Fill và sau đó UnlockBits và thực hiện hàm Paint. Màu được tô sẽ được chọn là màu bên trái.

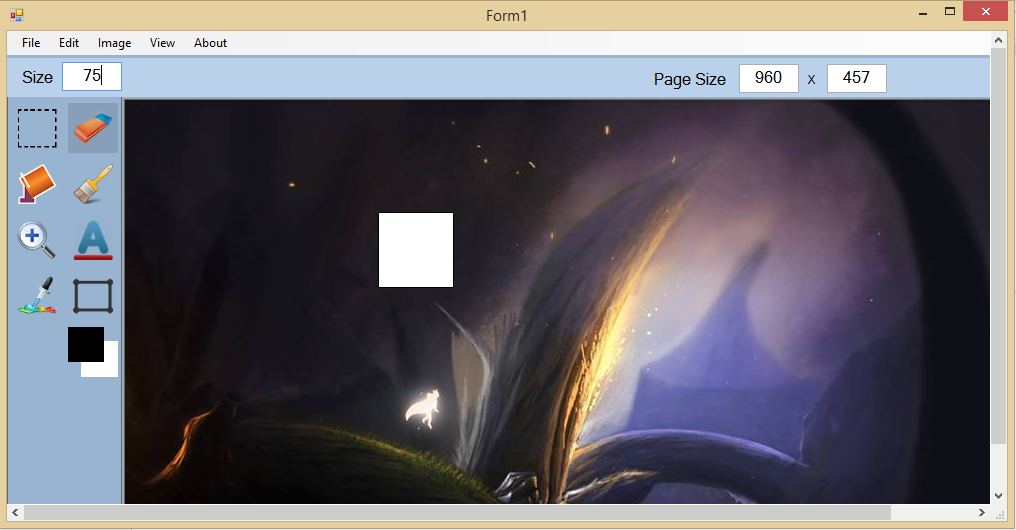


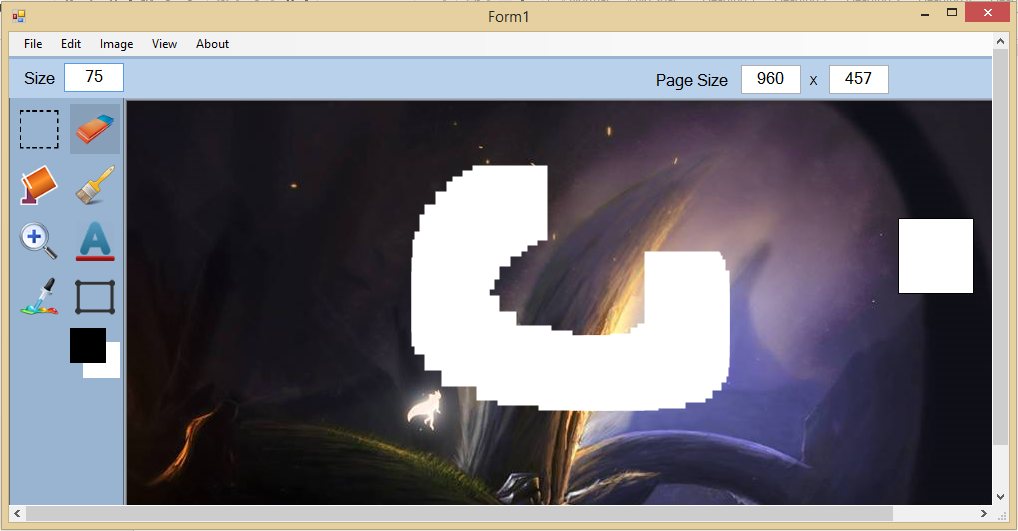
### **Eraser**

- Sơ đồ hoạt động



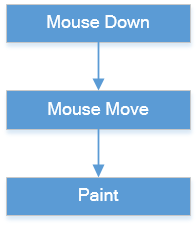
- Tiếp nhận sự kiện Mouse Down lấy tọa độ chuột, kiểm tra xem nếu mà lúc sự kiện Mouse Move diễn ra và sự kiện Mouse Down vẫn xảy ra thì sẽ cho tẩy theo đường đi của chuột, còn nếu chỉ nhấp thả thì xóa đi vùng được chọn.



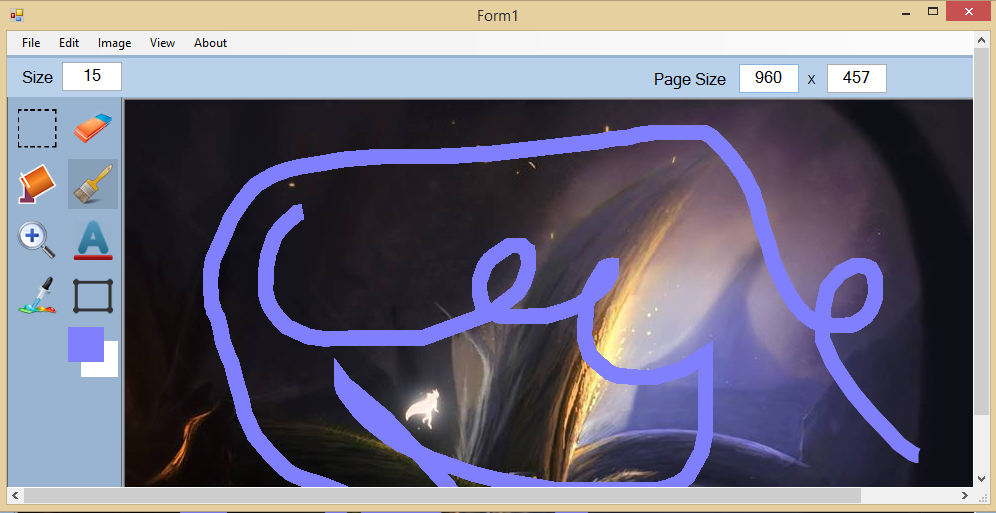


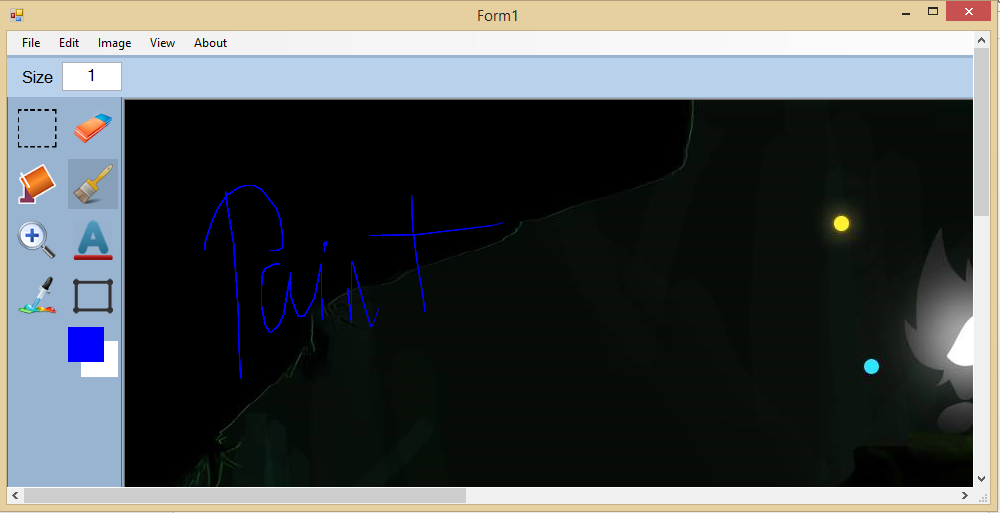
### **Brush**

- Sơ đồ hoạt động



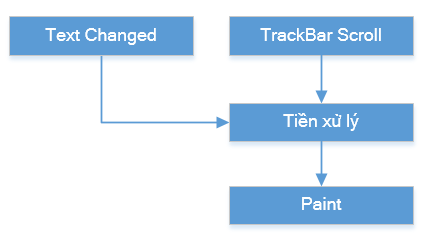
- Tiếp nhận sự kiện Mouse Down lấy tọa độ chuột, kiểm tra xem nếu mà lúc sự kiện Mouse Move diễn ra và sự kiện Mouse Down vẫn xảy ra thì sẽ thực hiện hàm DrawLine() nối các đường thằng lại thông qua các tọa độ chuột lấy được ở sự kiện Mouse Move. Ta phải dùng hàm DrawLine bởi vì tốc độ xử lý chậm hơn tốc độ nhận được tọa độ chuột nên nếu ta lướt chuột nhanh thì sẽ xảy ra hiện tượng mất điểm.



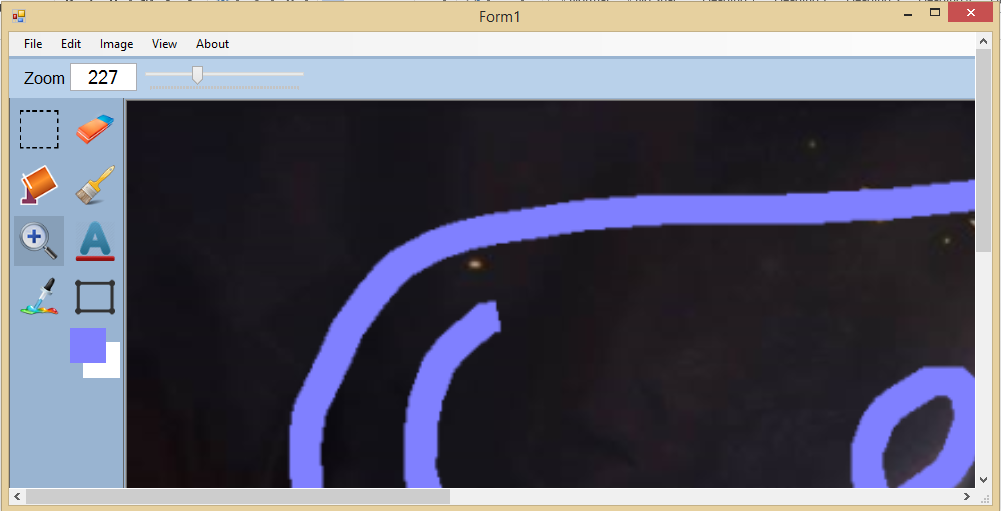


### **Zoom**

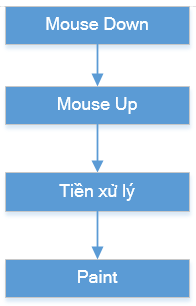
- Sơ đồ hoạt động



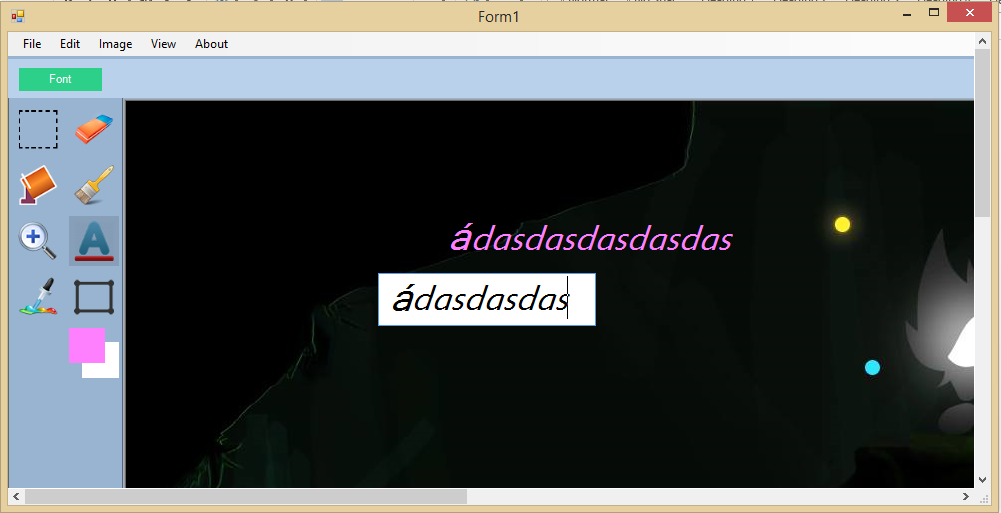
- Zoom không sử dụng các sự kiện chuột để thực hiện, sử dụng TrackBar Scroll để xác định tỉ lệ phóng to lên hoặc là nhập số vào TextBox. Lúc này sẽ tạo 1 Bitmap mới nhưng có kích cỡ to hơn, độ to theo đúng tỉ lệ ở TextBox. Và chỉ cho phép phóng to ảnh.



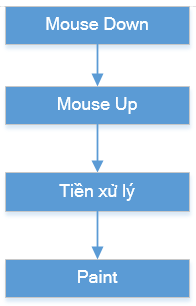
### **Text**



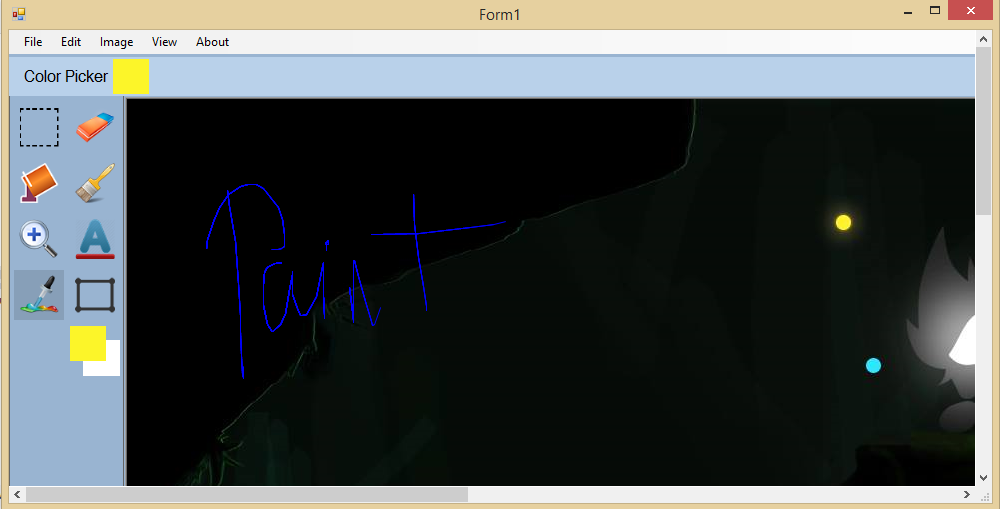
- Nhận sự kiện Mouse Up lấy tọa độ chuột, cho phép hiện TextBox lên và nhập Text vào đây đúng với Font được chọn, sau khi nhập xong nhấp chuột ra ngoài lúc này sẽ thực hiện hàm DrawString để vẽ Text lên Bitmap với Font được chọn.



### **Color Picker**

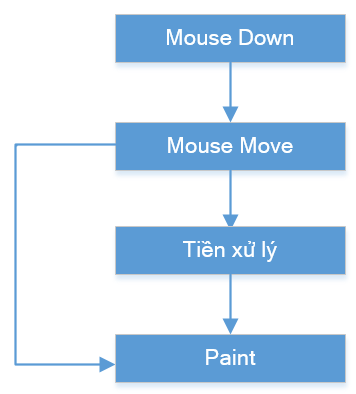


- Nhận sự kiện Mouse Up và lấy tọa độ chuột sau đó lấy màu tương ứng với tọa độ pixel trên Bitmap lúc này màu được chọn sẽ gán cho LeftColor, màu được chọn khi nhấp chuột trái.

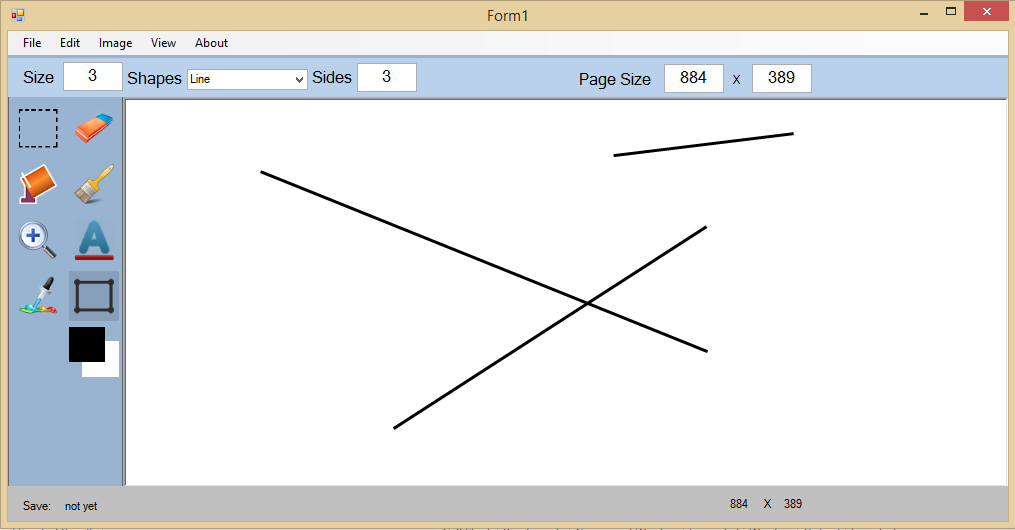


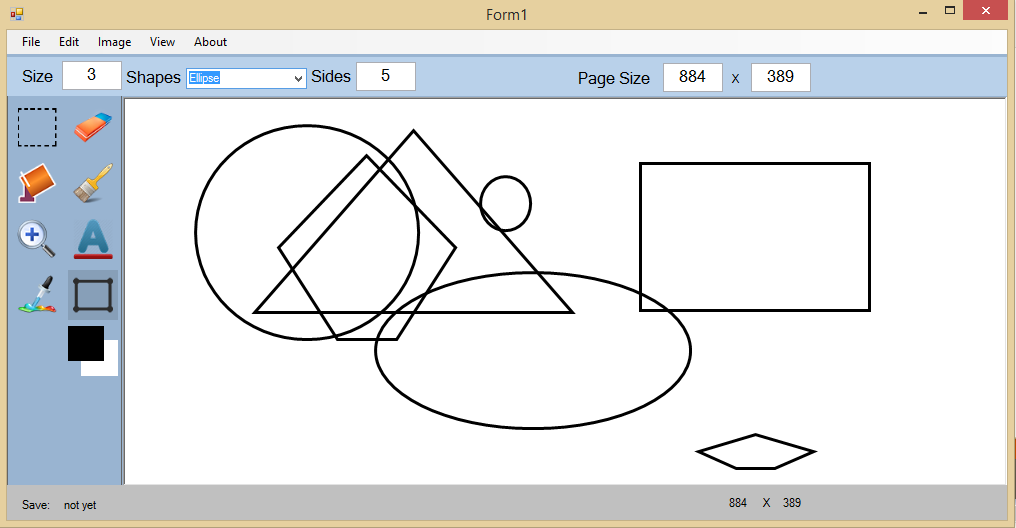
### **Shape**

- Sơ đồ hoạt dộng



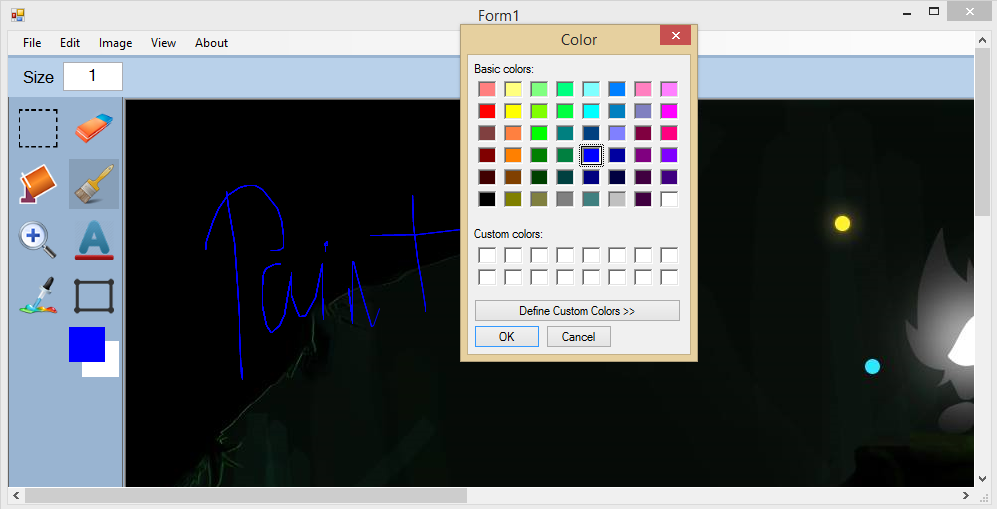
- Tiếp nhận sự kiện Mouse Down lấy tọa độ chuột, kiểm tra xem nếu mà lúc sự kiện Mouse Move diễn ra và sự kiện Mouse Down vẫn xảy ra thì sẽ thực hiện hàm vẽ với hình được chọn (DrawLine, DrawPolygon, DrawEclipse,….) trong lúc Mouse Move và Mouse Down cùng xảy ra thì sẽ gọi hàm Paint để vẽ lên 1 hình mô phỏng hình đang được vẽ và khi nhận sự kiện Mouse Up thì sẽ kết thúc quá trình vẽ và vẽ hình lên Bitmap.





### **5.9 Colors**

- Hiện ColorDialog khi nhấp vào để chọn màu cho LeftColor và RightColor.



## **Kĩ thuật**

### **Thuật toán Flood Fill**

- là thuật toán dùng tô màu được dùng cho chức năng fill.

- Nguyên lý hoạt động:

* + Lấy 1 điểm gốc ban đầu đó là điểm nhấp chuột xuống vùng tô màu và nhét vào Stack.
  + Tiếp đến ta quét các pixel phía bên trên đến khi gặp pixel màu khác với màu nền hoặc là chạm tới cạnh bên trên của Bitmap.
  + Ta bắt đầu tô màu các pixel từ trên xuống dưới cho đến khi gặp pixel khác màu mới màu nền hoặc chạm cạnh dưới của Bitmap. Trong quá trình tô màu xuống bên dưới thì ta sẽ kiểm tra phía bên trái và bên phải của pixel đang tô màu có giống màu nền không (tức là chưa được tô) nếu giống thì ta sẽ thêm vào Stack, và chỉ lấy 1 lần bên trái và 1 lần bên phải (dùng 2 biến bool để xác định xem đã lấy chưa, một biến cho bên trái và 1 biến cho bên phải).
  + Sau khi tô xong đường thẳng đầu tiên thì ta tiếp tục lấy điểm tiếp theo trong Stack và thực hiện lại, vòng lặp sẽ kết thúc khi số phần tử trong Stack hết.

### **Lock bits**

- Là kĩ thuật dùng để cải tiến việc Get/Set màu của pixel.

- Trong C# nếu chúng ta truy xuất để lấy màu và tô màu cho pixel trong 1 Bitmap lớn nhiều lần và liên tục sẽ gây ra hiện tượng đơ chương trình nếu máy yếu có thể bị crash.

- Nguyên lý hoạt động:

* Đầu tiên ta sẽ cần một đối tượng BitmapData và khóa vùng cần xử lý của Bitmap lại thông qua phương thức LockBits.
* Sau khi lấy được BitmapData ta sẽ dùng 1 con trỏ int (IntPtr) để lấy pixel đầu tiền trong BitmapData.
* Tiếp đến ta sẽ lấy độ dài của của ma trận pixel, ở đây ma trận là mảng 1 chiều, BitmapData có hỗ trợ thuộc tính Stride = Width \* 4 (4 ở đây là vì 1 pixel có 4 thuộc tính là ARGB).
* Sau khi có độ dài ma trận pixel của Bitmap ta tạo 1 mảng kiểu byte sau đó copy dữ liệu từ BitmapData sang ma trận này thông qua con trỏ Int đã lấy ở trên, ta dùng hàm System.Runtime.InteropServices.Marshal.Copy của C#.
* Mọi việc xử lý sẽ được thực hiện trên ma trận kiểu byte trên.
* Sau khi xử lý xong thì ta UnlockBits bằng cách copy lại dữ liệu từ mảng kiểu byte sang vị trí con trỏ int và dùng phương thức UnlockBits của Bitmap với tham số truyền vào là BitmapData.

# **III – TỔNG KẾT**

## **Đánh giá**

* Phần mềm chạy tốt trên các máy hỗ trợ .NET framework 3.5 trở lên.
* Kích thước chương trình nhỏ gọn (hơn 100KB)
* Chiếm tài nguyên máy ít (khoảng 10 – 15MB RAM)
* Tốc độ xử lý chưa tốt lắm, vẫn còn giật, lag đặc biệt với những ảnh có kích thước lớn
* Vẫn có khả năng có bugs nhưng chưa tìm ra.

## **Website và tài liệu tham khảo**

**- Websites:**

* + Thuật toán tô màu:

<http://vietgamedev.net/forum/thread/206/help-gi%E1%BA%A3i-thu%E1%BA%ADt-t%C3%B4-m%C3%A0u-loang/>

* + Tham khảo các thư viện của C#:  
    http://stackoverflow.com, <https://msdn.microsoft.com> và cùng các website khác.