# Technical Code Guide

**Index:**

**Capture Routine:**

There are two parts to my application that deal with the capture routine. The capture routine of the web camera and capture routine of the coordinates of the hand.

The capture routine of the web camera sets up a capture window and uses the grab frame function to capture frames from the camera. Note that the preview rate is set to zero. When initially setting up the capture window. Instead of having the API function signal the capture of a frame this application uses a timer that calls capGrabFrameNoStop functions to signal the FrameCallback function to be called. The FrameCallback function is called when a frame is received. This was used because timers run when the application does not have focus unlike the API callback function, which does not continue to receive frames when the application loses focus. Look to the code for further details on this process. The capture routine also uses a coverts.dll provided by Professor Blahnik that allows for the video picture to be build in a bit map and then transferred at once to the picture display.

The other part of my capture routine deals with my capturing of the hands coordinates. I created a function called FindLargestObject, which looks for the largest object in the frame of the camera. To supports this functions I created a class call HandObject. Every time a new object is found in my view a new instance of the HandObject class is created. This class holds values such as the size, topmost coordinates, the leftmost coordinates, and the rightmost coordinates. This class also gives stack support by having a pop and push routine of pixel coordinates. These functions are used by the FindLargestObject function in my algorithm to find the largest object. Note also by finding the largest object in the frame I can ignore other small objects. This makes the program less light sensitive. The FindLargest Object function contains a routine that is recursive in nature, but has been written in a non-recursive manner. The HandObject stack support of the class supports this algorithm and allowed me to remove the recursion. I chose to do this because I had learned that when working with a recursive routine in VB it is very easy to blow the internal stack. The algorithm is very simplistic in nature it starts at one side of the object and continues to go down, right, and then up in a recursive nature. Every pixel that is found is pushed on the stack and then if the object gets stuck it pops the last pixel off the stack and then tries the other direction. Note that each pixel that is found is changed to a number above the color spectrum of 255 to signal a stopping condition for the routine. Note my algorithm only works if you start at the left top most coordinate of the object. The following is a c++ view of my HandObject class and pictures to show a progression of how the algorithm works.

**Class HandObject:**

**Private:**

int leftx - the leftmost pixels x coordinate
int lefty - the leftmost pixels y coordinate
int topx - the top pixels x coordinate
int topy - the top pixels y coordinate
int rightx - the rightmost pixels x coordinate
int righty - the rightmost pixels y coordinate
long Count - the size of the cordx and cordy arrays
int cordx() - the array that holds the x coordinates
int cordy() - the array the holds the y coordinates
long amount - the size of the object

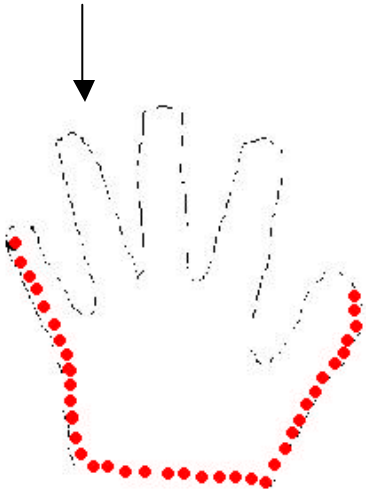
**Public:**

Class_Initialize() – initializes the class

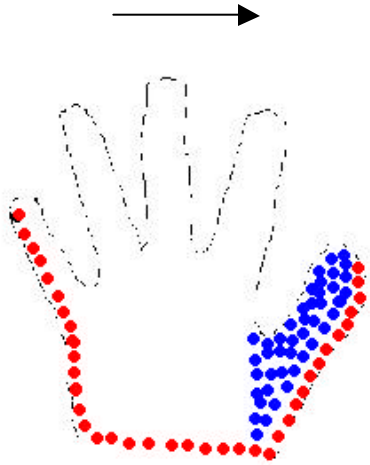
//Assessors
void settopy( int stopy)
void setleftx (int sleftx)
void setlefy(int slefty)
void setrightx(int srightx)
void setrighty(int srighty)
long Size()
long CountSize()
int SpreadLeftx()
int SpreadLefty()
int SpreadRighty()
int SpreadRightx()
int SpreadTopx()
int SpreadTopy()

void Push( int x, int y) - stores a pixel coordinate in the array
void Pop(int & x, int & y) - retrieves the pixel coordinate from the array

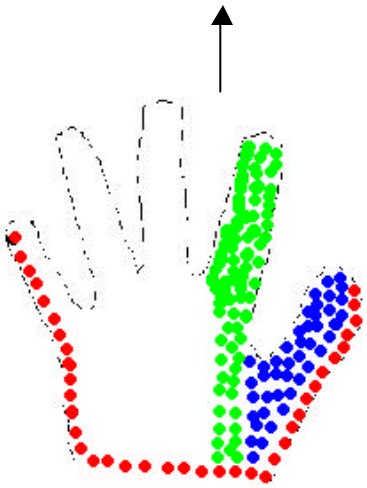
Go **down** as much as you can. If you can't down or right then go to the last coordinate and try to go down

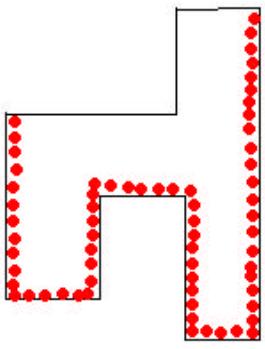


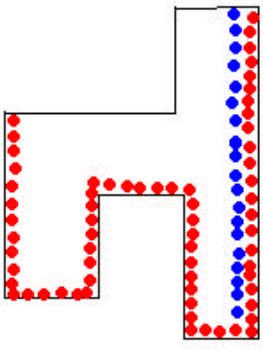
Go **right** as much as you can. . If you cant go right or up then go to the last coordinate and try to go right.

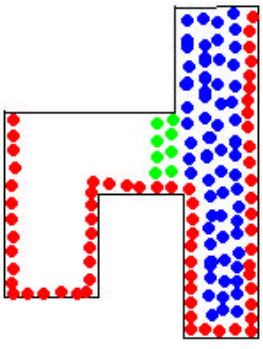


Go **up** as much as you can. If you cant go up go to the last coordinate and try to go up.









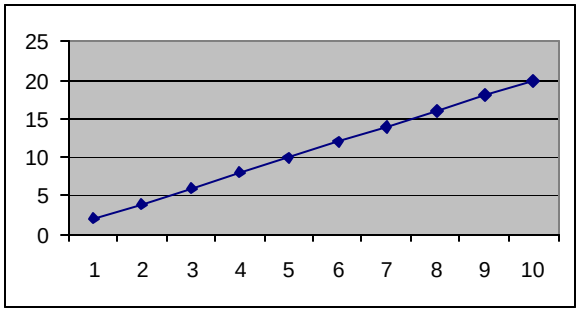
**Calibration/Initialization:**

The application contains an Initialization function, which is located in the main Mouse Properties Form. This function calls the FindLargestObject to get the coordinates of the hand. This function then uses these coordinates to calculate the leftspread, rightspread, and height. Note this function also adjusts it capture routine for left handed user.

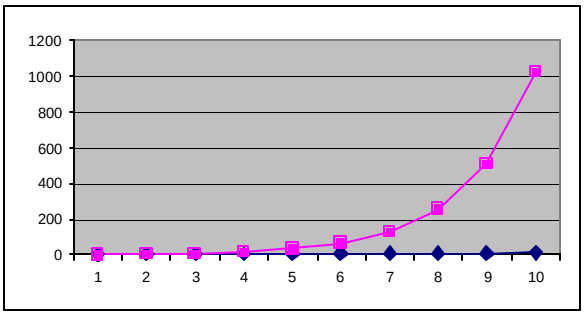
**Movement Routine**

There are two movement routines in the application the linear and expediential. The movement functions are located in the MouseApiFunctions module. The linear movement routine moves the object as many times in relation to the movement sensitivity setting. For example if the object moved 20 pixels and the linear sensitivity was set to 2 then the object would move 40 pixel. The expediential movement routine moves the mouse by squaring the distance to move. For example if the pixel change was 3 then the mouse would move 9 pixels.

Linear Movement

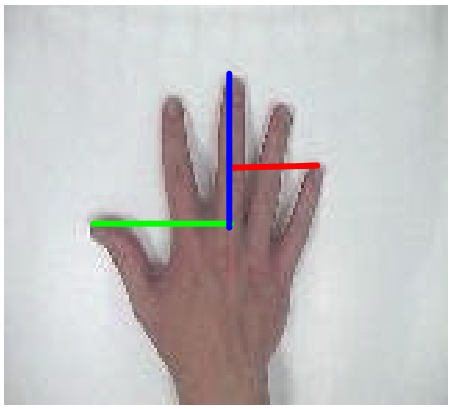


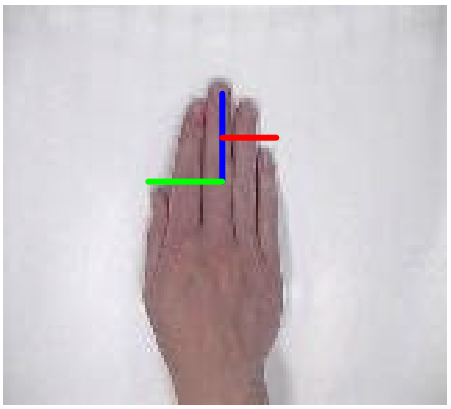

Expediential Movement




There is also a buffer setting in the application. This buffer setting prevents the movement algorithms from kicking in until the mouse has moved more than the buffer setting. This was necessary and useful to allow the mouse to reach every pixel on the screen and make the mouse movements more precise.

The movement routines only get called if the hand's left and right spread is proportionally smaller than the calibration spreads. So if the hand is closed the mouse will move if the hand is spread the mouse will not move. The spread difference is measured in percentages so if the camera is mounted higher or lower the application should not be affect. Unfortunately the higher the camera is mounted the more sensitive the application will be.





**Clicking:**

The clicking API functions are all located in the MouseApiFunctions. I have implemented three clicking option the left click, the right click, and the left down/up click. All these click are calculated based a combinations of the left spread, right spread, and height. Theses if statements are located HandleFrame functions. Note the user can only click if the object size is at lest 50% the size of the original calibration object. This is useful to prevent unwanted clicking or movement when entering the cameras view with your hand.

Left Click

Right Click

Mouse Down/
Mouse Up

**Save Routine:**

The save routine allows the use to save there personal setting and also allows the setting of the user to be save when the application terminates. There are two files involved in this process iniVMM file and the mousesettings.txt file. The iniVMM holds the last settings of the application. The mousesettings file holds all the saved settings of the user. These two files are read in when the program is loaded and written out when the application is closed.

IniVMM File Format Example:

Name
Location
Left spread
Right Spread
Light Setting
Left or Right handed
Height
Size of the Object
Buffer Size
Linear sensitivity
Motion styles
Click sensitivity

mousesettings File Format Example:

Name
Location
Left spread
Right Spread
Light Setting
Left or Right handed
Height
Size of the Object
Buffer Size
Linear sensitivity
Motion styles
Click sensitivity
Name
Location
Left spread
Right Spread
Light Setting
Left or Right handed
Height
Size of the Object
Buffer Size
Linear sensitivity
Motion styles
Click sensitivity
…….

**System Tray Code:**

The application uses API calls to allow the application to run in the system tray. All the system tray API calls are located in the SystemTrayApiFunctions module in my code. This module also contains the functions to add and delete a system tray icon for your application. There is also a function called MouseMoveCheck that can be called in the move mouse routine of the application, which allows for the application to be hidden or maximized when you click on the icon in the system tray.

**Process Priority**

Since my application needs to be very responsive I chosen to adjust my process priority to high. All my code to adjust the process priority is located in the ProcessPriorityApiFunctions module in my code. This module contains functions to set the process priority and to get the application current process priority.