



CARNIVORE

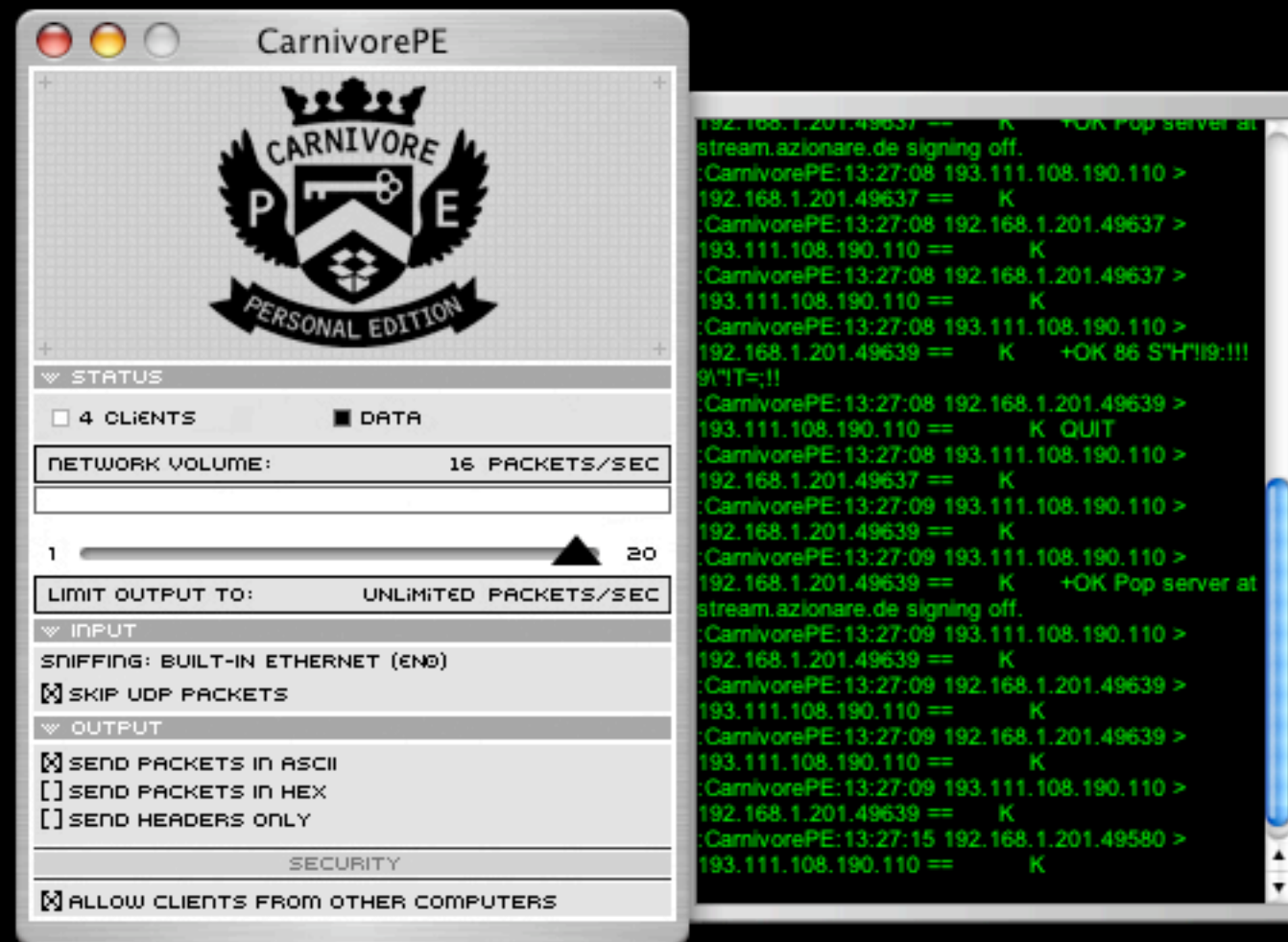
1. AN ANIMAL THAT EATS MEAT
2. NICKNAME DSC1000 (FBI)

WEBCAM

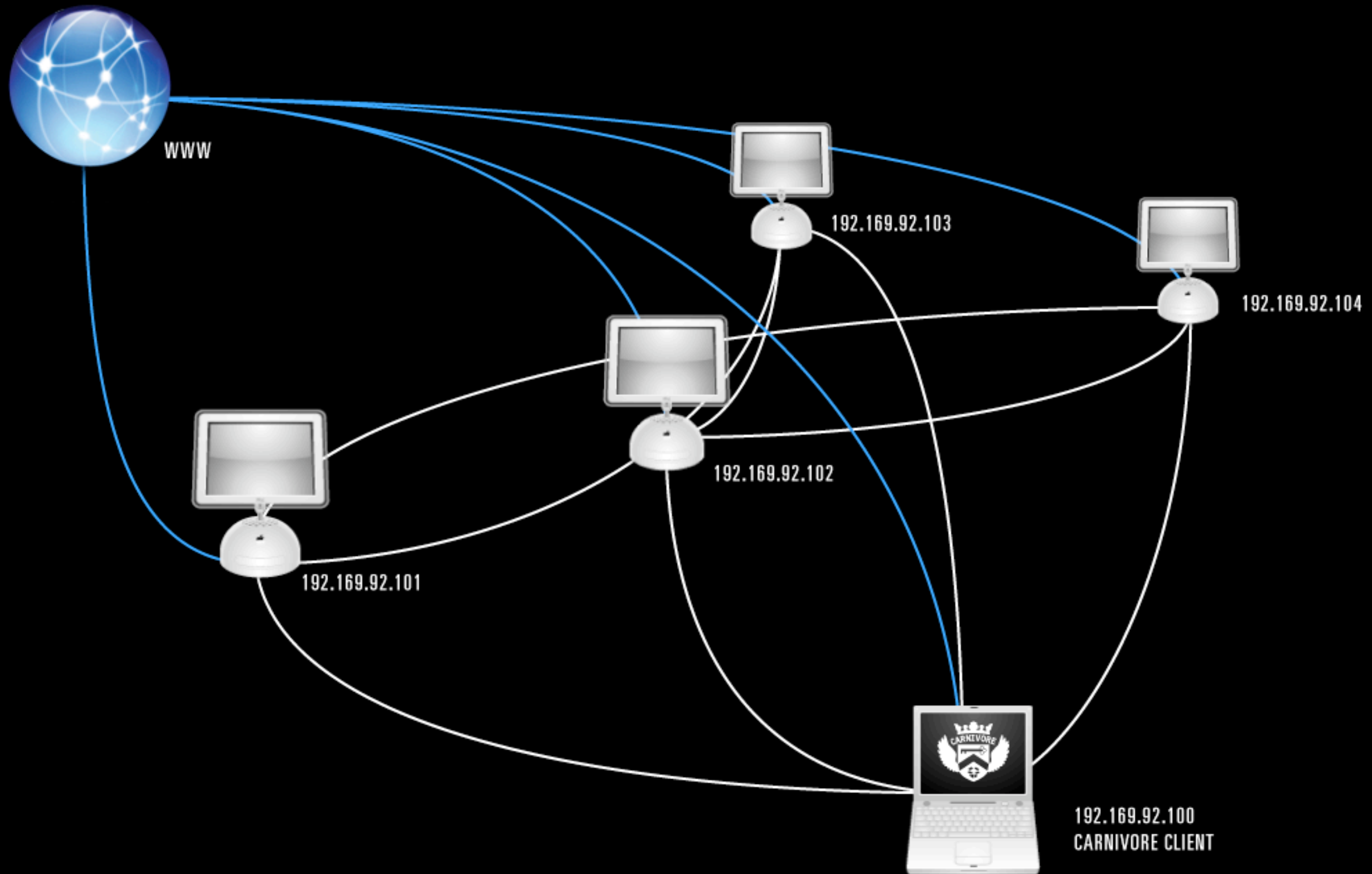
1. ABBREVIATION OF 'WEB CAMERA'
2. OPTICAL SURVEILLANCE TOOL

SNIFFER / SNIFFING

1. A POLICE DOG
2. NETWORK SURVEILLANCE TOOLS



<http://rhizome.org/carnivore/>



PORTS MAC OSX

<http://www.iana.org/assignments/port-numbers>
<http://support.stat.ucla.edu/view.php?supportid=39>

20	FTP Data	510	FirstClass server	5498	Hotline Tracker
21	FTP Control	515	LPR (printing)	5500	Hotline Server
22	SSH (secure shell remote login)	548	AFP (AppleShare)	5501	Hotline Server
23	Telnet	554	RTSP (QuickTime server)	6346	Gnutella
25	SMTP (email)	591	FileMaker Pro Web	6699	Napster/Macster client
53	DNS	626	IMAP Admin	7070	Real Player
70	Gopher	631	IPP (Internet Printing Protocol)	7648	CuSeeMe (video)
79	Finger	636	LDAP over TLS/SSL (LDAPS)	7649	CuSeeMe (video)
80	HTTP (Web)	660	ASIP Remote Admin	8080	Common HTTP alternate
88	Kerberos	666	Now Contact Server	16080	Web Performance Cache
105	PH (directory)	687	ASIP Shared UG Port	19813	4D server
106	Poppass (change password)	1080	WebSTAR Admin	etc...	
110	POP3 (email)	1417	Timbuktu Control (pre-5.2)		
111	Remote Procedure Call (RPC)	1418	Timbuktu Observe (pre-5.2)		
113	AUTH	1419	Timbuktu Send Files (pre-5.2)		
119	NNTP (News)	1420	Timbuktu Exchange (pre-5.2)		
139	NETBIOS Session	1443	WebSTAR/SSL Admin		
143	IMAP (new email)	3031	Program Linking (Apple Events)		
311	AppleShare Web Admin	3659	Apple SASL		
384	ARNS (tunneling)	4000	Now Public Event Server		
387	AURP (tunneling)	4199	EIMS Admin		
389	LDAP (directory)	4347	LANsurveyor Responders		
407	Timbuktu 5.2 or later	5003	FileMaker Pro		
427	SLP (service location)	5190	AOL Instant Messenger		
443	HTTP over SSL (HTTPS)				
445	MS Directory Service				
465	SMTP over SSL (SSMTP)				
497	Retrospect				

80	HTML	
53	DOMAIN	
110/25	POP3/SMTP	
5190	ICQ/AIM	
21	FTP	
6699/6346	NAPSTER/GNUTELLA	
	UNKNOWN	

:CARNIVOREPE:13:26:02 192.168.1.101.110 > 192.168.1.100.110:

HALLO DANIEL

LEIDER JA, HATTE EIN PAAR KLINGGLOTTEN IN DEUTSCHLAND HOCH REPARIEREN LASSEN, DANACH WAR DAS GERÄTESE ÜBER DEM SUPERHIVE HERBODEN, DIE INSTANZ KLEMMT, DER TOUCHPAD BUTTON MIT NEH WACKLER, USB-PORT LÖCKER, ETC.

HEUT SCHICK ICHS IHN. HOFFENTLICH KOMMST WER, WIEDER.

GEHEN DENN DIE JAHRE, WENN DU SIE BEIERS TERMINAL AUFRUFST?

FEIN, DASS DU DEN WIRD TRACKER NUTZEN KANNST, ICH FINDS IMMER GUT, WENN MAN NICHT ALLES VOM GRUNDAUF NEU PROGRAMMIEREN MUSS, DAS KOSTET SO VIEL ZEIT.

ICH FANGE SCHNELL AN MEIN HINDSARUNG PROJECT IN JAVA JZME FÜR'S HANDEY ZU REALISIEREN, DAS IST GAR NICHT SO KOMPLIZIERT, LEIDER GIBTS NOCH KEINEN PORT FÜR PROZESSING DAHER, ALLES LIESSE SICH ABER AUCH NICHT REALISIEREN, DIE DINGER SIND SCHON NUR SO LANGSAM.

GRÜßE

MICHAEL Z

:CARNIVOREPE:13:26:02 192.168.1.101.110 > 192.168.1.100.110:

HEADER

TIMESTAMP

SENDER IP

PORT

TO

RECEIVER IP

PORT

HALLO DANIEL

LEIDER JA. HATTE EIN PAAR KLEINIGKEITEN IN DEUTSCHLAND NOCH REPARIEREN LASSEN. DANACH WAR DAS GERÄTESE ÜBER DEM SUPERORIVE HERBODEN. DIE INSTANZEN KLEMMEN. DER TOUCHPAD BUTTON MIT NEH WÄNDLER. USB-PORT LÖCKER, ETC.

HEUT SCHICK ICWS IHL. HOFFENTLICH KOMMST WIE, WIEDER.

GEHEN DENN DIE JAHR. WENN DU SIE BEIHEB TERMINAL AUFRUFST?

FEIN, DASS DU DEN WIRD TRACKER NUTZEN KANNST. ICH FINDE IMMER GUT. WENN MAN NICHT ALLES VOM GRUNDAUF NEU PROGRAMMIEREN MUSS. DAS KOSTET SO WIEL ZEIT.

ICH FANGE SCHNELL AN MEIN HINDSARUNG PROJECT IN JAVA. JZME FREIS HANDE ZU REALISIEREN. DAS IST GAR NICHT SO KOMPLIZIERT. LEIDER GIBTS NOCH KEINEN PORT FÜR PROZESSOR DAFÜR. ALLES LIESSE SICH ABER AUCH NICHT REALISIEREN. DIE DINGER SIND SCHON NICH SAU LANGSAM.

GRÜßE

MICHAEL Z



13:26:02 192.168.1.101

TIMESTAMP SENDER IP



192.168.1.100

RECEIVER IP

HALLO DANIEL

LEIDER JA, HATTE EIN PAAR KLINGGLOTTEN IN DEUTSCHLAND HOCH REPARIEREN LASSEN, DANACH WAR DAS GERÄTESE ÜBER DEM SUPERHIVE HERBODEN, DIE INSTANTAN KLEMMT, DER TOUCHPAD BUTTON AM NEH WACKLER, USB-PORT LOCKER, ETC.

HEUT SCHICK ICHS IHN. HOFFENTLICH KOMMST WIE, WIEDER.

GEHEN DENN DIE JAHR, WENN DU SIE BEIERS TERMINAL AUFRUFST?

FEIN, DASS DU DEN WIRD TRACKER NUTZEN KANNST, ICH FINDS IMMER GUT, WENN MAN NICHT ALLES VOM GRUNDAUF NEU PROGRAMMIEREN MUSS, DAS KOSTET SO VIEL ZEIT.

ICH FANGE SCHNELL AN MEIN HINDSARING PROJEKT IN JAVA JZME FÜR'S HANDEY ZU REALISIEREN, DAS IST GAR NICHT SO KOMPLIZIERT, LEIDER GIBTS HOCH KEINEN PORT VON PROCESSEND RAUF, ALLES LIESSE SICH ABER AUCH NICHT REALISIEREN, DIE DINGER SIND SCHON NICH SAU LANGSAM.

GRÜßE

MICHAEL Z

```
// CARNIVORE WEBCAM SMIFFING ----- // // A CARNIVORE CLIENT // 2004 DANIEL ROTHHAUG, MAJID: ROTHHAUG@ZUMKLUCKUCK.COM // // MODIFIED 2004-07-07 // PS VERSION: 60 // REQUIRES SONIA VERSION: 2_0 // DIMENSIONS OF VIDEO-INPUT (WEBCAM, // BEST PERFORMANCE WITH 160X120 OR SMALLER INT VIDWIDTH = 320/4; INT VIDEHEIGHT = 240/4; // THE DIMENSIONS OF THE STAGE INT STAGewidth = 640; INT STAGEHEIGHT = 480; // WEBCAM TRACKER ----- TRACKER T: INT THRESHOLD = 20; FLOAT DIFF; BOOLEAN NEWFRAME = FALSE; BOOLEAN BGSET = FALSE; // CARNIVORE CLIENT ----- SOCKET SOCKET; BUFFEREDREADER PACKETREADER; CARNIVORELISTENER CL; CARNIVOREPARSER PARSER = NEW CARNIVOREPARSER(); // IP NUMBERS 0. DANIEL 1. AL 2. PERRY 3. PC 4. PHILLON STRING[] IPLIST = {"192.168.1.201", "192.168.1.203", "192.168.1.204", "192.168.1.213", "192.174.87.187"}; STRING IP = IPLIST[0]; STRING[] PLIST = NEW STRING[0]; // STORES THE PARSED PACKAGE STRING PACKET: STRING[] WORDS = NEW STRING[10000]; STRING[][] WORDLIST = NEW STRING[VIDWIDTH][VIDHEIGHT]; // PARTICLE SYSTEM ----- MYPARTICLE[] P = NEW MYPARTICLE[STAGewidth][STAGEHEIGHT]; BFONT MYFONT, MYSMALLFONT; BIMAGE ICON, MASK, GLOW, DIFFIMAGE, BGIMAGEBG, BGIMAGE, BGBLUR; CONVOLVER C; // SOME COLORS COLOR[] COLORS = {#660000, #6600FF, #C00000, #C00000, #FF0000, #000000, #000000, #000000}; // SOME ICONS STRING[] IMAGES = {"MAIL.GIF", "CHAT.GIF", "FTP.GIF", "HTTP.GIF", "HTML.GIF", "UNKNOWN.GIF", "PEERTOPEER.GIF"}; STRING[] MASKS = {"MAIL_MASK.GIF", "CHAT_MASK.GIF", "FTP_MASK.GIF", "HTTP_MASK.GIF", "HTML_MASK.GIF", "UNKNOWN_MASK.GIF", "PEERTOPEER_MASK.GIF"}; INT ICORNR = 0; // DEFAULT ICON INT COLORNR = 0; // DEFAULT COLOR INT MODE = 1; // DEFAULT DISPLAY-MODE INT RES = 1; // DISPLAY RESOLUTION BOOLEAN MOUSE = FALSE; BOOLEAN SHOWBACKGROUND = FALSE; BOOLEAN DISPLAY = TRUE; BOOLEAN PERSPECTIVE = FALSE; // X, Y Z ROTATION FLOAT XR = 0; FLOAT YR = 0; FLOAT ZR = 0; // SOUND SAMPLE IN: SAMPLE OUT // SETUP ----- VOID SETUP() { PRINTLN(SYSTEM.GETPROPERTY("JAVA.VM.VERSION")); SIZE[STAGewidth, STAGEHEIGHT]; SONIA.START(THIS); IN = NEW SAMPLE("TINK.AIFF"); OUT = NEW SAMPLE("BOTTLE.AIFF"); ICON = LOADIMAGE(IMAGES[ICORNR]); MASK = LOADIMAGE(MASKS[ICORNR]); GLOW = LOADIMAGE("GLOW.GIF"); ICON.ALPHA[MASK]; GLOW.ALPHA[GLOW]; MYFONT = LOADFONT("UNIVERS_50_ULTRA_CONDENSED.VFW"); MYSMALLFONT = LOADFONT("UNIVERS_50_ULTRA_CONDENSED_24.VFW"); // START WEBCAM TRACKER BEGINVIDE[VIDWIDTH, VIDEHEIGHT, 25; T = NEW TRACKER(); TRACKER[THRESHOLD, VIDWIDTH, VIDEHEIGHT]; DIFF = FLOOR[WIDTH/VIDWIDTH]; // START LISTENING TO CARNIVORE CL = NEW CARNIVORELISTENER(); CL.STARTLISTENING(); // SETUP PARTICLES FOR[INT I=0; I<VIDWIDTH; I++] { FOR[INT J=0; J<VIDHEIGHT; J++] { P[I][J] = NEW MYPARTICLE(I, J); } } // FILL WORDS WITH DUMMIES FOR[INT I=0; I<WORDS.LENGTH; I++] { WORDS[I] = STRING[INF, 2]; } // FILL PLIST WITH DUMMIES FOR[INT I=0; I<PLIST.LENGTH; I++] { PLIST[I] = " "; } // LOOP ----- VOID LOOP() { CLEAR(); BACKGROUND(); IF (PERSPECTIVE) { ROTATE[XR]; ROTATE[YR]; ROTATE[ZR]; } PACKET = CL.GETPACKET(); // GET A NEW PACKET IF (BGSET == TRUE) { IF (SHOWBACKGROUND && BBLURRED) { PUSH(); IMAGE[0][CORNER]; TMT[255, 100]; //MOTMT[0; IMAGE[BGBLUR, 0, 0]; POP(); } IF (NEWFRAME == TRUE) { T.GETPIXELDIFF[VIDEO, BGIMAGE, DIFF]; NEWFRAME = FALSE; } IF (MODE == 6) { INT[] RECTPOINTS = T.GETPIXELDIFF[VIDEO, BGIMAGE, DIFF]; NOFILL(); STROKE[255, 0, 0]; RECT[RECTPOINTS[0], RECTPOINTS[1], (RECTPOINTS[2]-RECTPOINTS[0]), (RECTPOINTS[3]-RECTPOINTS[1])]; } FOR[INT I=0; I<RECTPOINTS.LENGTH; I++] { PRINTLN("RECTPOINT" + "=" + I + "=" + RES) + " " + RECTPOINTS[I]; } } ELSE IMAGE[VIDEO, 0, 0]; PARSER.UPDATE(); //SAVEFRAME(); // SET BIG BACKGROUND IMAGE VOID BACKGROUNDIMAGE() ENVIDE[0; BEGINVIDE[640, 480, 25; DELAY[1000]; BGBLUR = "GETBACKGROUND[VIDEO];" BEGINVIDE[VIDWIDTH, VIDEHEIGHT, BOOLEAN BBLURRED="TRUE" COLORIMAGE="TRUE" PUBLIC BIMAGE GETBACKGROUND[BIMAGE REFBC] REFERENZBILD ANLEGEN UND ZURUECKGEBEN IMG=NEW BIMAGE[WIDTH, HEIGHT]; SYSTEM.ARRAYCOPY[REFBC.PIXELS, IMG.PIXELS, REFBC.PIXELS.LENGTH-1]; C="FADEALPHA[MYCOLOR," CONVOLVER[1]; C.SETRADIUS[2]; C.BLUR[IMG, 0, 0, WIDTH, HEIGHT]; IF (COLORIMAGE) FOR (INT I<(WIDTH*HEIGHT)-WIDTH; INT C="GREEN[CURRENT];" IMG.PIXELS[I] = "COLOR[0][5/2, G/2, 2];" RETURN IMG; PARTICLE ----- CLASS MYPARTICLE FLOAT X, Y, Z, TX, TY, HOME_X, HOME_Y, RX, RY, SIZE, BRIGHT, F="30F;" FRICTION E="10F;" ELASTICITY STRING MYTEXT; W; COLOR C; _X, _Y) DEFAULT POSITION BX="RANDOM[WIDTH]" (RANDOM[500] - RANDOM[500]); BY="RANDOM[HEIGHT]" TRACKED X=_X * DIFF; Y=_Y TX="X" TY="Y" HOME_X="X" HOME_Y="Y" W="INT[RANDOM[WORDS.LENGTH-1];" BENDER[] ELLIPSWIDE[CENTER_DIAMETER]; IMAGEMODE[CENTER_DIAMETER]; TEXTMODE[CENTER_DIAMETER]; #MODE="-" 1 || MODE="-" 2 3) TRANSLATE[0, 0, 2]; TRANSLATE[0, 0, 2]; B="BLUE[CURRENT];" 55; { $ > .1) SIZE=BRIGHT {S*4+8; SWITCH #MODE CASE 1: TMT[0; IMAGE[GLOW, SIZE]; BREAK; 2: 3: TMT[255, 255, 255, 0; IMAGELCON, 4; FILL[RED, C, GREEN, C, BLUE, C]; 0; 0; 7; 0; 0; 0; FILL[255; NOSTROKE]; RECT[TX, DIFF-1, DIFF-1; //////////////////////////////////////////////////// GOOD PIXELS TRACK[COLOR MYCOLOR] BRIGHT="BRIGHTNESS[MYCOLOR]/255;" MOVE[HOME_X, 4] {DIFF<="120;" SCALE[DIFF]; SCALE[120]; BENDER[0; NASTY HIDE[COLOR [MOUSE] MOVE[MOUSEX, MOUSEY, MOVE[BY, COLOR[255, 255, 255, 0]; 1; 255, SCALE[0.01]; ELASTIC MOVEMENT NX, NY, NS, NZ; MOVE[FLOAT TARGET_X, TARGET_Y, TARGET_Z] NX=NX [TARGET_X X]; NY=NY [TARGET_Y Y] NZ=NZ [TARGET_Z Z] Z SCALING S="1;" SCALE[FLOAT TARGET_S] NS=NS 0.1 [TARGET_S S] 0.3; FADE TR, TG, TB; FADECOLOR[COLOR CURRENT, TARGET] B="RED[CURRENT];" TR=0.1 F [RED[TARGET] B] 0.3F; TG=TG [GREEN[TARGET] G] TB [BLUE[TARGET] B] COLOR[B, G, ALPHA, TA; FADEALPHA[COLOR FROM, TO] A="FROM;" TA=TA 0.30F [TO A] 0.60F; 0, A); CARNIVORE PACKET="CL.GETPACKET[0]; PARSER ----- CARNIVORE-PARSER LASTPACKET; TIME="NF[0000, 0;"; SENDERIP="SENDER[0]; SENDERPORT="SENDER[0]; SENDERPORTNAME="PORT[SENDER[0]; RECEIVERIP="RECEIVER[0]; RECEIVERPORT="RECEIVER[0]; RECEIVERPORTNAME="PORT[RECEIVER[0]; CONTENT="" DEBUG="TRUE;" CARNIVOREPARSER[] GET NEW FROM CLIENT UPDATE[0; LASTPACKET I="0" && PACKETINDEXOF[P] PARSER]; LASTPACKET="PACKET" [DISPLAY] TRANSLATE PORTS-NUMBERS PORT[STRING PORTNR] [PORTNR.EQUALS["0"]; "HTTP:" [PORTNR.EQUALS["53"]; "DOMAIN:" [PORTNR.EQUALS["110"]; "POP3:" [PORTNR.EQUALS["25"]; "SMTP:" [PORTNR.EQUALS["143"]; "IMAP:" [PORTNR.EQUALS["110"]; "ICM/AIM:" [PORTNR.EQUALS["6346"]; "GMAIL:" [PORTNR.EQUALS["6688"]; "MAPSTER:" [PORTNR.EQUALS["548"]; "APPLESHARE:" "UNKNOWN:" ICON="LOADIMAGE[IMAGES[ICORNR]]; AND COLORSTYLE SETSTYLE[STRING ICORNR="G" COLORNR="G" [PORTNR.EQUALS["21"]; MASK="LOADIMAGE[MASKS[ICORNR]]; ICON.ALPHA[MASK]; SETCOLOR[STRING COLORS[4]; COLORS[3]; COLORS[0]; COLORS[1]; COLORS[2]; COLORS[6]; COLORS[5]; //STRING[] HISTORY=NEW STRING[10000]; //INT LINES="0;" PARSE PACKETS PARSE[] MAKE SHURE THERE IS NO SPACE CHARACTER IN FRONT OF IT. _PACKET="PACKET.TRIM[0; SPLIT HEADER STRING[] TEMP="SPLIT[PACKET " " " " WHITESPACE; 2; NF[0000, 0; UHR; SENDER="SPLIT[TEMP[1], " "; " " SENDER[1] SENDER[2] SENDER[3]; RECEIVER="SPLIT[TEMP[3], " "; " " RECEIVER[1] RECEIVER[2] RECEIVER[3]; ONLY CONTAINS CONTENT. (TEMPLENGTH 0) L=TEMPLENGTH 4; I<TEMPLENGTH " ; TEMPWORDS="SPLIT[CONTENT]; FOR[INT I<WORDS.LENGTH; WORDS[I]="TEMPWORDS[INT[RANDOM[TEMPWORDS.LENGTH]]; PLIST[0]="TIME;" PLIST[1]="SENDERIP;" PLIST[2]="SENDERPORT" PLIST[3]="SENDERPORTNAME" PLIST[4]="RECEIVERIP;" PLIST[5]="RECEIVERPORT" PLIST[6]="RECEIVERPORTNAME" PLIST[7]="CONTENE" //PLIST[8]="PACKET" //SETSTYLE[PLIST[2]; CHECK INCOMING OR OUTGOING CHANGE PORT VISUALISATION [REQUALS[PLIST[2]; SETSTYLE[PLIST[2]; OUTPLAY[0; PRINTLN("> OUT PLIST[2]; SETSTYLE[PLIST[5]; INPLAY[0; PRINTLN("< IN PLIST[5];); _RES="CONTENTLENGTH[0; " / _RES < 50 RES="1;" 100] DIFF="FLOOR[WIDTH/(VIDWIDTH*RES)];" SAVE TO FILE [LINES HISTORY.LENGTH] HISTORY[LINES]="JOIN[PLIST, " "; ; SAVESTRINGS["HISTORY.TXT", HISTORY]; (DEBUG) PRINTLN("W" "NEW IP @ PACKETINDEXOF[P]); PRINTLN("LENGTH _RES; PRINTLN("TEMP :\t" TEMP[0]; I<SENDER.LENGTH; PRINTLN("SENDER SENDER[0]; I<RECEIVER.LENGTH; PRINTLN("RECEIVER RECEIVER[0]); PRINTLN("TIME :\t" +TIME); PRINTLN("SENDER / ( " "); PRINTLN("RECEIVER: PRINTLN("CONTENT:CONTENT); LINESPACE LS="20;" FILL[255, 255, 255, 30; RECT[0, 0, WIDTH, 50; DISPLAY TEXT [PACKETLENGTH] TEXTFONT[MYSMALLFONT, 24; TEXT[PLIST[0], 20, LS*4]; FILL[SETCOLOR[PLIST[2];]; TEXT[PLIST[1] PLIST[3], 100, FILL[SETCOLOR[PLIST[5];]; TEXT[PLIST[4] PLIST[6], 200, TEXT[PLIST[7], LS*2]; WEBCAM TRACKER ----- ORIGINAL "MIGTRACKER" BY MICHAEL ZIEGLER MAJID:ZIEGLER@FORMWERKS.DE PART THE PROJECT "DEMOGRAPHIC DATA AUGMENTED REALITY" HTTP://DIGITALEINFORMATIONARCHITEKTUR.MIG.DE RECURRENT, GOUBRENT, GOUBRENT, REF, GREE, BREF, YMIN, XMIN, ROFF, GOFF, BOFF, TEMPMG; XMAX="0;" YMAX="0;" TRACKER[INT TOLERANCE, QUADWIDTH, QUADHEIGHT] THIS.CURRTOLERANCE="TOLERANCE;" THIS.OW="QUADWIDTH;" THIS.OH="QUADHEIGHT;" XMIN="STAGewidth;" YMIN="STAGEHEIGHT;" GETBACKGROUND[BIMAGE BGC=NEW BIMAGE[REFBC, WIDTH, REFBC, HEIGHT]; BGC.PIXELS, BGC; INT[] GETPIXELDIFF[BIMAGE CURRENTFRAME, BGC, DIFF] VERÄNDERUNGEN FINDEN VIDEHEIGHT; J="J+RES" VIDWIDTH; RECURRENT="VIDEO.PIXELS[*VIDWIDTH+J]>> 16] & 0xFF; GOUBRENT = (VIDEO.PIXELS[*VIDWIDTH+J] >> 8) & 0xFF; GOUBRENT = VIDEO.PIXELS[*VIDWIDTH+J] & 0xFF; BREF = (BGC.PIXELS[*VIDWIDTH+J] >> 16) & 0xFF; GREF = (BGC.PIXELS[*VIDWIDTH+J] >> 8) & 0xFF; BREF = BGC.PIXELS[*VIDWIDTH+J] & 0xFF; ROFF = BREF - RECURRENT; GOFF = GREF - GOUBRENT; BOFF = BREF - GOUBRENT; INT X = INT[*DIFF]; INT Y = INT[*DIFF]; COLOR C = COLOR[RECURRENT, GOUBRENT, GOUBRENT]; IF (ROFF > CURRTOLERANCE || GOFF > CURRTOLERANCE || BOFF > CURRTOLERANCE) { //////////////////////////////////////////////////// // TRACK GOOD PIXELS P[0][0].TRACK[C]; IF (X < XMIN) XMIN = X; IF (Y < YMIN) YMIN = Y; IF (X > XMAX) XMAX = X; IF (Y > YMAX) YMAX = Y } ELSE { //XMIN = -1; //////////////////////////////////////////////////// // HIDE BAD PIXELS P[0][0].HIDE[C]; } } INT[] CHANGEDRECT = {XMIN, YMIN, XMAX, YMAX}; RETURN CHANGEDRECT } PUBLIC INT CURRTOLERANCE, OW, OH; } // CARNIVORE ----- // // A CARNIVORE CLIENT FOR PROCESSING, ADAPTED FROM MARK NAPIER'S CPLD JAVA CODE // COPYRIGHT (C) APRIL 2004 BY RSG CLASS CARNIVORELISTENER IMPLEMENTS RUNNABLE { // FOR CONNECTING TO SOCKET STRING HOST = "127.0.0.1"; INT PORT = 6667; //STREAM STUFF SOCKET S; DATAINPUTSTREAM IN; PRINTSTREAM OUT; BOOLEAN CONNECTED = FALSE; BOOLEAN STOPTHREAD = FALSE; // RUNS THE LISTENER THREAD RUNNER; // PACKET STUFF STRING _PACKET = " ; //////////////////////////////////////////////////// // START THE CARNI LISTENER WITH THE OUTPUT PANEL AS PARAM. // SET DEFAULT CONNECTION METHOD TO CARNIVORE SERVER. PUBLIC CARNIVORELISTENER() { //////////////////////////////////////////////////// // START/STOP THE LISTENING LOOP PUBLIC VOID STARTLISTENING() { // START THE REPAINT THREAD RUNNER = NEW THREAD[THIS]; RUNNER.START(); STOPTHREAD = FALSE; } PUBLIC VOID STOPLISTENING() { RUNNER = NULL; STOPTHREAD = TRUE; DISCONNECTFROMSERVER(); } //////////////////////////////////////////////////// // CONNECT TO THE DATA SOURCE PUBLIC BOOLEAN CONNECTTOSEVER() { CONNECTED = FALSE; // DEFAULT OPEN SOCKET MESSAGE("CONNECTING TO HOST " + HOST + " " + PORT + " "); TRY { // CONNECT TO SERVER SOCKET S = NEW SOCKET[HOST, PORT]; // TO READ FROM SERVER IN = NEW DATAINPUTSTREAM[S].GETINPUTSTREAM(); // TO SEND TO SERVER OUT = NEW PRINTSTREAM[S].GETOUTPUTSTREAM(); MESSAGE("CONNECTED TO " + HOST + " " + PORT); CONNECTED = TRUE; } CATCH (EXCEPTION E) { MESSAGE("CONNECT[0; " + E); CONNECTED = FALSE; } RETURN CONNECTED; } PUBLIC VOID DISCONNECTFROMSERVER() { MESSAGE("DISCONNECT"); TRY { IF (OUT != NULL) { OUT.PRINTLN("QUIT"); OUT.CLOSE(); } IF (IN != NULL) { IN.CLOSE(); } } CATCH (EXCEPTION E) { MESSAGE("ERROR WHEN CLOSING INPUTSTREAM: " + E); } CONNECTED = FALSE; } PUBLIC STRING GETPACKET() { RETURN _PACKET; } //////////////////////////////////////////////////// // THREAD RUNS IN TWO STATES // 1) LOOP UNTIL CONNECTED TO SERVER // 2) READ LINES UNTIL THREAD STOPS OR SOCKET FAILS // IF SOCKET FAILS, RETURN TO STATE 1. PUBLIC VOID RUN() { STRING LINE; CONNECTED = FALSE; MESSAGE("RUN"); WHILE {STOPTHREAD==FALSE} { GET CONNECTION WHILE {CONNECTED==FALSE && STOPTHREAD==FALSE} { IF {CONNECTTOSEVER() == FALSE} { TRY { THREAD.SLEEP[6000]; } RETRY IN 60 SECS } CATCH (INTERRUPTEDEXCEPTION E) { MESSAGE("INTERRUPTED: " + E); } } } // READ LINES FROM CARNI SERVER TRY { BYTE BB=0; WHILE {CONNECTED==TRUE && STOPTHREAD==FALSE} { LINE = IN.READLINE(); IF (LINE == NULL) { BREAK; } _PACKET = LINE; // YIELD A LITTLE TIME FOR REPAINT OPERATIONS TO WORK THREAD.SLEEP[100]; // 100 } } CATCH (EXCEPTION E) { MESSAGE("ERROR WHEN LISTENING TO " + HOST + " " + PORT + " " + E); } FINALLY { MESSAGE("CONNECTION CLOSED BY SERVER."); DISCONNECTFROMSERVER(); // WILL LOOP BACK TO CONNECTTOSEVER() TO RETRY CONNECTION EVERY 10 SECS } } } PUBLIC VOID MESSAGE[STRING MSG) { SYSTEM.OUT.PRINTLN("CARNIVORELISTENER: " + MSG; ) } // FAST GAUSSIAN BLUR V1.3 ----- // BY MARIO KLINGEMANN<HTTP://INDICATOR.OASIMONDO.COM>CLASS CONVOLVER[INT RADIUS; INT KERNELSIZE; INT[] KERNEL; INT[]] MIKE CONVOLVER[INT SZ] THIS.SETRADIUS[SZ]; VOID SETRADIUS[INT SZ] INT L; SZ=MIN[MAX[1, SZ], 248]; IF (RADIUS==SZ) RETURN; KERNELSIZE=1+SZ*2; RADIUS=SZ; KERNEL = NEW INT[1+SZ*2]; MULT=NEW INT[1+SZ*2][256]; INT SUM=0; FOR (I=1; I<SZ; I++) INT SZ=SZ-1; KERNEL[SZ+I]=KERNEL[SZ]-SZ*SZ; SUM+=KERNEL[SZ]+KERNEL[SZ]; FOR (J=0; J<256; J++) MULT[SZ+I][J]=MULT[SZ][J]-KERNEL[SZ]*J; } KERNEL[SZ]=SZ*SZ; SUM+=KERNEL[SZ]; FOR (J=0; J<256; J++) { MULT[SZ][J]=KERNEL[SZ]*J; } VOID BLUR[BIMAGE IMG, INT X, INT Y] INT VOUT, INT H[ (INT SUM, CR, CG, CB, K; INT PIXEL_READ, L, R, X, Y, YL, YL, YL, RW; INT) PIX=IMG.PIXELS; INT IW=IMG.WIDTH; INT WH=IMG.HEIGHT; INT RQ=NEW INT[WH]; INT GQ=NEW INT[WH]; INT BQ=NEW INT[WH]; FOR (I=0; I<WH; I++) { RI="POX[0; BQ=" [RI&AMP;0XFF0000] >> 16; GQ=" [RI&X00FF00] >> 8; BQ=" [RI&X0000FF]; } INT R2[0; NEW INT[WH]; INT G2[0; NEW INT[WH]; INT B2[0; NEW INT[WH]; X=MAX[0, X]; Y=MAX[0, Y]; W=X+W-MAX[0, (X+W)-RW]; H=Y+H-MAX[0, (Y+H)-RW; HEIGHT]; YL=Y*RW; FOR (YL=Y+YL<H; YL++) { FOR (XL=X; XL<W; XL++) { CB=CG=CR=SUM=0; RI=XL-RADIUS; FOR (I=0; I<KERNELSIZE; I++) READ="RI+" { IF (READ>=X && READ<W) { READ+=YL CR+=MULT[I][R[READ]; CG+=MULT[I][G[READ]; CB+=MULT[I][B[READ]; SUM+=KERNEL[I]; } RI=Y+XL; R2[0; CR/SUM; G2[0; CG/SUM; B2[0; CB/SUM; } YL+=RW; } YL=Y*RW; FOR (YL=Y+YL<H; YL++) { YL=YL-RADIUS; RW=Y*RW; FOR (XL=X; XL<W; XL++) { CB=CG=CR=SUM=0; RI=YL; READ=XL+RW; FOR (I=0; I<KERNELSIZE; I++) { IF (RI<H && RI>=Y) { CR+=MULT[I][R2[READ]; CG+=MULT[I][G2[READ]; CB+=MULT[I][B2[READ]; SUM+=KERNEL[I]; } RI++; READ+=RW; } PIX[XL+YL]=0XFF00000 + (CR/SUM)<<16 + (CG/SUM)<<8 + (CB/SUM); } YL+=RW; } } } // KEYBOARD CONTROLS ----- VOID KEYPRESSED() { SWITCH (KEY) { // SET BACKGROUND IMAGE CASE ' ': BGIMAGE = T.GETBACKGROUND[VIDEO]; BGSET = TRUE; BREAK; CASE 'X': BACKGROUNDIMAGE(); BREAK; // INCREASE THRESHOLD CASE '+': THRESHOLD += 2; PRINTLN("THRESHOLD:\t" + THRESHOLD); BREAK; // DECREASE THRESHOLD CASE '-': THRESHOLD -= 2; PRINTLN("THRESHOLD" + THRESHOLD); BREAK; // CHANGE VISUALISATIONS CASE '1': MODE = 1; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '2': MODE = 2; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '3': MODE = 3; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '4': MODE = 4; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '5': MODE = 5; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '6': MODE = 6; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '7': MODE = 7; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '8': MODE = 8; PRINTLN("VISUALISATION:\t" + MODE); BREAK; CASE '9': MODE = 9; PRINTLN("VISUALISATION:\t" + MODE); BREAK; // INCREASE RESOLUTION CASE ': RES += 1; PRINTLN("RESOLUTION:\t" + RES); DIFF = FLOOR[WIDTH/VIDWIDTH]; BREAK; // DECREASE RESOLUTION CASE ';': IF (RES > 1) { RES -= 1; PRINTLN("RESOLUTION:\t" + RES); DIFF = FLOOR[WIDTH/VIDWIDTH]; } BREAK; // CHANGE PORT VISUALISATION CASE 'N': IF (ICORNR < IMAGES.LENGTH-1) { ICORNR += 1; COLORNR += 1; } ELSE { ICORNR = 0; COLORNR = 0; } ICON = LOADIMAGE(IMAGES[ICORNR]); MASK = LOADIMAGE(MASKS[ICORNR]); ICON.ALPHA[MASK]; BREAK; // 3D ROTATION CASE RIGHT YR += 0.036; PERSPECTIVE = TRUE; BREAK; CASE LEFT YR -= 0.036; PERSPECTIVE = TRUE; BREAK; CASE UP XR += 0.036; PERSPECTIVE = TRUE; BREAK; CASE DOWN XR -= 0.036; PERSPECTIVE = TRUE; BREAK; // SHOW/HIDE THE DISPLAY CASE 'D': DISPLAY = !DISPLAY; BREAK; // SHOW/HIDE BACKGROUND IMAGE CASE 'B': SHOWBACKGROUND = !SHOWBACKGROUND; BREAK; // ENABLE/DISABLE MOUSE ATTRACTOR CASE 'M': MOUSE = !MOUSE; BREAK; // SAVE SCREENSHOT CASE 'S': SAVEFRAME[YEAR] + " " + MONTH] + " " + DAY] + " " + HOUR] + " " + MINUTE] + " " + SECOND] + " " + CARNICAM.TGA); PRINTLN("SCREENSHOT SAVED"); BREAK; } } VOID VIDEVENT() { NEWFRAME = TRUE; } PUBLIC VOID STOP() { SONIA.STOP(); SUPER.STOP(); }
```

