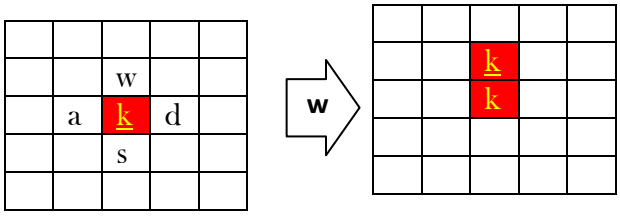
PAINT your LIFE

Paint Your Life è un programma ideato e compilato a cura di Alessandro Cacco tra febbraio e maggio 2012. Ha avuto 3 versioni ufficiali: *1.0, 1.2 e 2.0*: il primo è limitato ad una area di disegno libero senza salvataggio né caricamento di lavori precedentemente svolti, il secondo permette di salvare il contenuto del disegno su file di tipo testo, ma veniva salvato solo il colore di ogni pixel senza lettere né colori corrispondenti, il terzo è la versione attuale, che permette il salvataggio di tutto il disegno, quindi pixel, lettera e colore della stessa, più il colore di sfondo (quello che quindi utilizza la gomma). È in programmazione la versione 2.2, che aggiunge a tutte le funzionalità di *PYL 2.0* menù contestuali ed un'area di disegno ampliata.

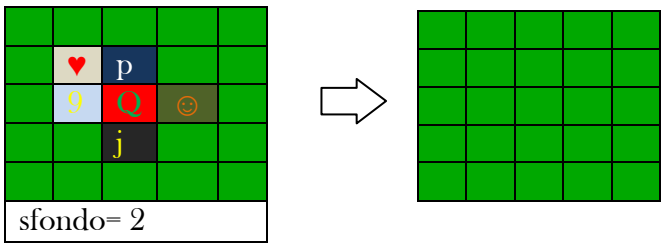
PYL è ispirato allo storico *Microsoft Paint*©, ma il funzionamento è leggermente diverso: innanzitutto, i comandi devono essere dati da tastiera, i file salvati sono di tipo *.pyl* e non possono essere creati dal programma di disegno (funzionalità che sarà fornita nella versione *2.2*).

Il programma sfrutta la ben nota funzione GOTOXY del modulo CRT del Pascal, disponibile nel compilatore *Bloodshed*© *Dev-Pascal,* e utilizza come tasti direzionali le classiche lettere A,S,D e W, con le quali ci si può spostare nell'area di disegno (se si va oltre il bordo, il cursore si sposta dalla parte opposta della stessa). I comandi sono questi:

- a,s,d,w = tasti direzionali

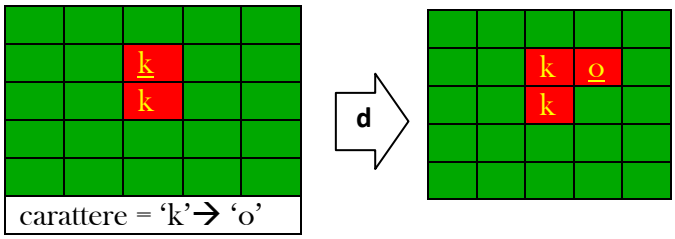


- r = cancella tutto

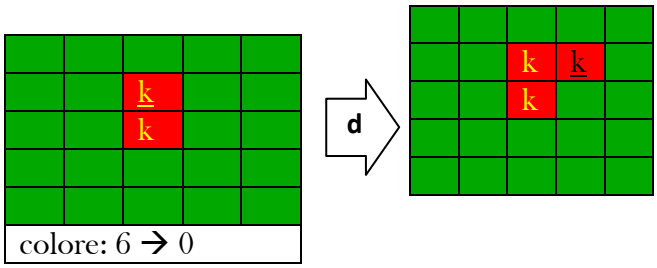


- b = cambia colore sfondo: se si cancella con la gomma o se si cancella tutto con *r*, i pixel diventeranno del colore scelto con questo comando

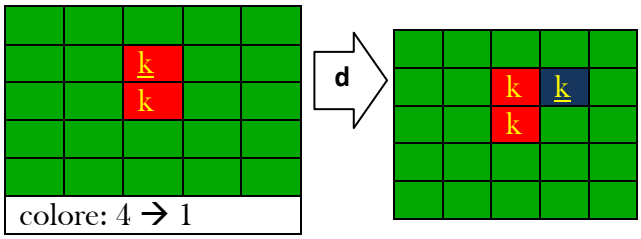
- l = cambia lettera



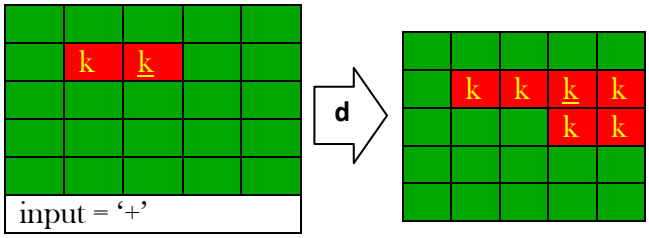
- t = cambia colore lettera



- c = cambia colore pixel (sfondo della lettera)



- -, + = cambia dimensione penna (1÷6)



- p = cambia modalità: scrittura con il pennarello, cancellazione con la gomma o spostamento neutro.

- q = uscita e salvataggio nel file pre-caricato, sovrascrivendo i vecchi dati

IL CODICE SORGENTE

1. *program PaintYonrLife;*
2. *uses crt;*

- DICHIARAZIONE DELLE VARIABILI -

3. var
x,j,k,y,i,penna,dim,background_color,writing_color,letter_color:integer;

Le variabili *i, j* e *k* serviranno a scorrere gli array tridimensionali (riga 6, 7)che di cui si parlerà in seguito .
x e *y* sono le coordinate del cursore, quindi dov'è posizionato e da dove si sposterà.
penna indicherà la modalità di spostamento, sia pennarello, gomma o neutra (2,1,0).
dim indica la dimensione della scrittura (se penna = 1 o 2).
Background_color (0÷7) conterrà il numero del colore dello sfondo, che 'colorerà' la gomma e di cui diventa tutta l'area dopo il comando *r*.
Writing_color (0÷7) conterrà il numero del colore dello sfondo dei pixel che verranno colorati col pennarello.
Letter_color (0÷7) conterrà il numero del colore con cui verrà scritta la lettera nei pixel colorati col pennarello.

4. c,letter:char;

Nella variabile *c* viene memorizzata la scelta dell'utente, quindi un comando od uno spostamento.
La variabile *letter* conterrà la lettera che sarà scritta in ogni pixel con il pennarello.

5. pixel:array[1..80]of array[1..24]of array[1..2]of integer;
6. pixletter:array[1..80]of array[1..24]of char;
7. bitmap:text;
8. percorso:string;

In *PYL* 2.0, i dati relativi ad ogni 'pixel' sono 3: colore sfondo, colore lettera e lettera (es. 3,2,'f'). Questi dati relativi ad ogni disegno si trovano in una variabile *text*, chiamata *bitmap*, assegnata ad un file *.pyl* che si trova dove è scritto nella variabile *percorso*. Questi numeri e lettere sono caricati in un array tridimensionale, *pixel*, ed in uno bidimensionale, *pixletter*. Si potrebbe pensare che metterli entrambi in un unico array tridimensionale sarebbe stato più comodo, però *pixletter* è di tipo *char*, poiché contiene le lettere di ogni 'cella' del disegno, mentre *pixel* è di tipo integer, poiché contiene tutti i colori dello sfondo di ogni cella (*pixel* [<coordinata x>[<coordinata y>][1]) ed i rispettivi colori dei caratteri (*pixel* [coordinata x][coordinata y][2]).

- IL PROGRAMMA -

```
9.    Begin
```

- TITOLAZIONE: UN PROGRAMMA ESTERNO -

```
10.       textbackground(white);
11.       clrscr;
12.       gotoxy(12,2);
13.       textbackground(0);
14.       write(' ');
15.       gotoxy(13,2);
16.       write(' ');
[…]
688.   textbackground(4);
689.   write(' ');
690.   gotoxy(59,23);
691.   write(' ');
692.   gotoxy(60,23);
693.   write(' ');
694.   gotoxy(61,23);
695.   write(' ');
```

In queste numerose righe viene scritto il titolo iniziale, la schermata iniziale che accoglie l'utente. Questa scrittura sembra molto lunga e faticosa da compilare, ma in realtà è stata scritta con un altro programma che converte i disegni creati con Paint Your Life 2.0 in istruzioni pascal pronte per essere compilate in un codice.

- APERTURA DEI DISEGNI: RICHIESTA E PREPARAZIONE ALL'USO -

```
696.   textbackground(white);
697.   textbackground(red);
698.   textcolor(white);
699.   gotoxy(20,13);
700.   write(chr(201));
701.   for i:=1 to 38 do write(chr(205));
702.   write(chr(187));
703.   gotoxy(20,14);
704.   write(chr(186));
705.   gotoxy(59,14);
706.    write(chr(186));
707.   gotoxy(20,15);
708.   write(chr(200));
709.   for i:=1 to 38 do write(chr(205));
710.   write(chr(188));
711.   gotoxy(21,14);
```

```
712.    read(percorso);
713.    percorso:=percorso+'.pyl';
```

In *PYL* 1.2 e successivi i file di disegno da caricare vengono richiesti all'utente in un rettangolo che compare al centro della schermata iniziale. Questi cicli *for* scrivono al centro della schermata un area con i caratteri delle tabelle e si posizionano lì dentro per far inserire all'utente il file da caricare. L'aggiunta dell'estensione *.pyl* è automatica da *PYL 2.0.,* quindi al contrario della prima versione dell'*1.2* non ammette l'uso di file di disegno *.txt.*

```
714.    assign(bitmap,percorso);
715.    reset(bitmap);
```

Il file localizzato nel percorso inserito dall'utente viene quindi assegnato alla variabile *bitmap*, di tipo *text* con il comando *assign.* Con il comando *reset* il cursore di *bitmap* viene posizionato all'inizio del testo in modalità lettura.

```
716.    for j:=1 to 2 do
717.        for i:=1 to 24 do
718.            for k:=1 to 80 do readln(bitmap,pixel[k][i][j]);
719.    for i:=1 to 24 do
720.        for k:=1 to 80 do readln(bitmap,pixletter[k][i]);
721.    readln(bitmap,background_color);
```

Viene quindi letto il file di testo, grazie ai due cicli *for* annidati, uno triplo ed uno doppio. Questi scorrono gli *array* sopra citati, memorizzando in ordine tutti i colori (in numeri) ed i caratteri. Come precedentemente detto, l'ultimo numero

- RIPRODUZIONE DEL DISEGNO SALVATO NELL'AREA -

```
722.    for i:=1 to 24 do
723.        for k:=1 to 80  do
724.            begin
725.                gotoxy(k,i);
726.                textbackground(pixel[k][i][1]);
727.                textcolor(pixel[k][i][2]);
728.                write(pixletter[k][i]);
729.            end;
```

Vengono quindi scritte tutte le celle del disegno nell'area, scorrendo allo stesso modo di prima i vettori, identificando le giuste coordinate e scrivendo i dati relativi ad ogni destinazione nella corretta posizione.

- I COMANDI E LE INFORMAZIONI DI DISEGNO -

```
730.    gotoxy(1,25);
```

```
731.    textbackground(white);
732.    textcolor(black);
733.    for i:=1 to 80 do write('-');
```

Ai piedi della schermata vengono scritti dei trattini (------)per la lunghezza totale della riga, cove poi verranno scritte sopra le informazioni per l'utente riguardo la modalità di scrittura, la grandezza di scrittura, il carattere scritto ed i colori utilizzati (gomma, penna e carattere).

```
734.    x:=1;
735.    y: =1;
736.    penna:=0;
737.    dim:=1;
738.    writing_color:=0;
739.    letter_color:=0;
740.    letter:=' ';
```

Qui vengono inizializzate le variabili che saranno utilizzate per il disegno: *x* e *y* serviranno come coordinate del cursore; *penna* indicherà la modalità di scrittura e varierà tra 0 e 2, 0 sarà il cursore neutro, 1 la penna e 2 la gomma. *dim* sarà in un range da 1 a 6 e sarà la dimensione di scrittura, cioè se la gomma od il pennarello scriveranno su un'area 1x1, 2x2, 3x3, 4x4, 5x5 o 6x6 celle; *writing_color* e *letter_color* indicheranno rispettivamente il colore di scrittura attuale del pixel e del carattere; *letter* sarà invece il carattere in uso nella scrittura a pennarello, poiché nella gomma viene scritto uno spazio bianco. Si parte dalle coordinate (1,1) con i colori neri, con uno spazio come lettera, con dimensioni di scrittura 1x1 ed in modalità neutra.

```
741.    repeat
742.        repeat
743.            if c='r' then
744.                begin
745.                    textbackground(background_color);
746.                    gotoxy(1,1);
747.                    for i:=1 to 80*24 do write(' ');
748.                    for i:=1 to 24 do
749.                        for k:=1 to 80 do pixel[k][i][1]:=background_color;
750.                    for i:=1 to 24 do
751.                        for k:=1 to 80 do pixletter[k][i]:=' ';
752.                    gotoxy(1,1);
753.                end;
```

Questa è la prima funzione verificata, se il tasto inserito dall'utente sarà 'r', allora l'area di disegno verrà riempita completamente da spazi bianchi del colore impostato come sfondo e i relativi array di informazioni verranno azzerati allo stesso modo. Poiché non è scritta alcuna lettera, ma uno spazio, è inutile azzerare anche il colore della lettera, che andrà sovrascritto non appena si andrà a disegnarci sopra.

```
754.        if c='b' then
755.            begin
756.                gotoxy(60,25);
757.                for i:=1 to 10 do write('-');
758.                gotoxy(60,25);
759.                repeat
760.                    read(background_color);
761.                until (background_color<=7)and(background_color>=0);
762.                gotoxy(1,1);
763.            end;
```

Se l'utente premerà 'b', sarà cambiato il colore della gomma, o di cui diventerà l'area dopo il comando 'r'. Vengono cancellati i tre vecchi colori dalla schermata e viene richiesto all'utente un numero fra 0 e 7, che sostituirà il vecchio colore di sfondo.

```
764.        textcolor(black);
765.        textbackground(white);
766.        gotoxy(1,25);
767.        write('carattere: ''');
768.        if c='l' then
769.            begin
770.                gotoxy(13,25);
771.                readln(letter);
772.                gotoxy(1,1);
773.            end;
774.        gotoxy(13,25);
775.        write(letter,'''-----');
```

Questo è il comando per cambiare il carattere in uso nella scrittura (dal pennarello), viene spostato il cursore sull'ultima riga e viene richiesto l'inserimento di un nuovo carattere che sovrascriverà quello vecchio memorizzandosi in *letter*.

```
776.        if c='t' then
777.            repeat
778.                gotoxy(60,25);
779.                for i:=1 to 10 do write('-');
780.                gotoxy(60,25);
781.                read(letter_color);
782.                gotoxy(1,1);
783.            until (letter_color<=7)and(writing_color>=0);
```

Con questo comando, analogamente al comando 'b', viene cambiato il colore della lettera in uso dal pennarello. La procedura è esattamente quella del cambiamento del colore di sfondo.

```
784.        if c='c' then
```

```
785.            repeat
786.                    gotoxy(60,25);
787.                    for i:=1 to 10 do write('-');
788.                    gotoxy(60,25);
789.                    read(writing_color);
790.                    gotoxy(1,1);
791.            until (writing_color<=7)and(writing_color>=0);
792.        gotoxy(60,25);
793.        write('----');
```

Questo, allo stesso modo dei comandi 'b' e 't', richiede l'inserimento di un nuovo colore si sfondo per i pixel disegnati col pennarello. Come prima, viene spostato il cursore all'ultima riga, il programma si mette in attesa di un nuovo colore (1÷7) e poi questo viene memorizzato in *writing_color*

```
794.        gotoxy(53,25);
795.        write('colore: ',background_color,', ',writing_color,', ',letter_color);
```

Queste righe riscrivono i colori in uso in una data posizione della riga 25. Se sono gli stessi di prima, l'utente non si accorgerà del cambiamento, se cambieranno invece questi compariranno sopra a dove l'utente aveva digitato il nuovo colore.

```
796.        if (c='+')and(dim<6)and(x<81-dim)and(y<25-dim) then dim:=dim+1;
797.        if (c='-')and(dim>1) then dim:=dim-1;
```

Con + e – cambierà la grandezza della scrittura, tra 1 e 6, ma se il cursore è troppo vicino al bordo la scrittura non si ingrandirà più di questo.

```
798.        if c='p' then penna:=penna+1;
799.        if penna=3 then penna:=0;
```

Se premuto il tasto 'p', cambierà la modalità di scrittura. Essa aumenterà da 0 a 2, quindi se diventerà 3 sarà decrementata a 0.

```
800.        for i:=30 to 45 do
801.            begin
802.                gotoxy(i,25);
803.                write('-');
804.            end;
805.        gotoxy(20,25);
806.        write('cursore: ');
807.        gotoxy(29,25);
808.        if penna=0 then write('neutro');
809.        if penna=1 then write('pennarello - ',dim);
810.        if penna=2 then write('gomma - ',dim);
```

Circa al centro della riga 25, verrà scritto *cursore:* <modalità di scrittura> - <dimensione>.

```
811.            gotoxy(x,y);
812.            c:=readkey;
813.        until (c='a')or(c='w')or(c='s')or(c='d')or(c='q');
```

Qui finisce il ciclo dei comandi. Le condizioni di uscita sono se *c* è una direzione o se è 'q', quindi scelta di uscita dal *PYL.*

```
814.        if c='a' then x:=x-1;
815.        if c='s' then y:=y+1;
816.        if c='d' then x:=x+1;
817.        if c='w' then y:=y-1;
818.        if x=0 then x:=81-dim;
819.        if x=82-dim then x:=1;
820.        if y=0 then y:=25-dim;
821.        if y=26-dim then y:=1;
```

Se l'utente digiterà uno dei tasti direzionali le variabili indicanti le coordinate si modificano, se supereranno i limiti verranno modificate nuovamente nel limite opposto.

```
822.        gotoxy(x,y);
823.        textcolor(letter_color);
```

Viene quindi spostato il cursore alle nuove coordinate e viene impostato il colore della lettera.

```
824.        if penna=1 then textbackground(writing_color);
825.        if penna=2 then textbackground(background_color);
```

Se la modalità di disegno corrente sarà *pennarello* (1) il pixel disegnato sarà del colore di scrittura, mentre se è in uso la *gomma* (2) sarà disegnato un pixel del colore scelto come sfondo.

```
826.        if penna<>0 then
827.            for i:=y to y-1+dim do
828.                for k:=x to x-1+dim do
829.                    begin
830.                        gotoxy(k,i);
831.                        if penna=1      then write(letter)
832.                                        else write(' ');
```

Se non sarà in uso la modalità *neutra* il cursore spostandosi scriverà in un area quadrata con i lati lunghi tanti pixel quanti il valore di *dim* il carattere scelto od uno spazio bianco, a seconda se si sta rispettivamente disegnando o cancellando.

```
833.                    if penna=1      then
834.                        begin
835.                            pixel[k][i][1]:=writing_color;
836.                            pixel[k][i][2]:=letter_color;
837.                            pixletter[k][i]:=letter;
838.                        end
839.                        else
840.                        begin
841.                            pixel[k][i][1]:=background_color;
842.                            pixel[k][i][2]:=letter_color;
843.                            pixletter[k][i]:=' ';
844.                        end;
845.                end;
```

vengono quindi memorizzati i nuovi pixel colorati negli array contenenti i dati di disegno, quindi il colore delle celle, il colore del carattere ed il carattere stesso (che sarà uno spazio se si starà cancellando. [i due *if … then … else* sono stati separati volontariamente per maggior chiarezza della spiegazione, nel programma coincidono].
Ogni memorizzazione negli *array* ed ogni spostamento verranno quindi ripetuti quante volte serve per modificare tutti i pixel richiesti dalla grandezza inserita.

```
846.        until (c='q');
```

Il programma terminerà con il comando 'q', che farà terminare il ciclo *repeat until* che si apriva verso l'inizio (riga 741)

```
847.        rewrite(bitmap);
```

Usciti dal ciclo, si salva automaticamente il file su quello originale (se si desidera terminare l'esecuzione del programma senza salvare è sufficiente cliccare sulla classica crocetta in alto a destra della finestra). Con il comando *rewrite* il cursore nel file di testo associato alla variabile *bitmap* (di tipo *text*) viene posizionato all'inizio della prima riga, pronto a cancellare e sovrascrivere tutto il testo presente precedentemente.

```
848.        for j:= 1 to 2 do
849.            for i:=1 to 24 do
850.                for k:=1 to 80 do writeln(bitmap,pixel[k][i][j]);
851.        for i:=1 to 24 do
852.            for k:=1 to 80 do writeln(bitmap,pixletter[k][i]);
```

Vengono scritti tutti gli array nel giusto ordine nel file, come spiegato precedentemente, con un numero o lettera per ogni riga. Ancora una volta viene utilizzato il ciclo *for*, poiché si sa a priori il numero di ripetizioni, coincidente all'ampiezza dell'area di disegno.

```
853.    writeln(bitmap,background_color);
```

Nell'ultima riga viene quindi scritto il colore utilizzato come sfondo (utilizzato dalla gomma e dal comando 'r') al momento della chiusura.

```
854.    close(bitmap);
855.  end.
```

Viene quindi chiuso il file di disegno associato a *bitmap* con il comando *close* e viene terminato il programma.