Merge char bullets with background chars

by Achim

Char bullets look quite ugly when flying around. A way to achieve a more aesthetic look is to generate new chars on the fly everytime the bullet's moving one char ahead.

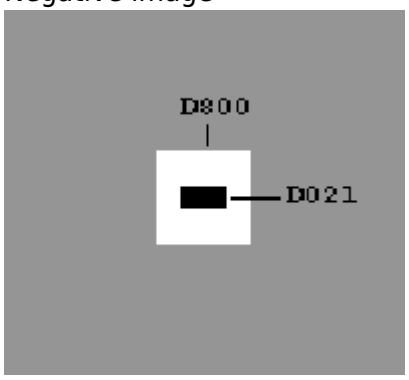
First thing to do is to reserve as many empty chars in your charset as you need bullets in your game.

For example 8 chars:

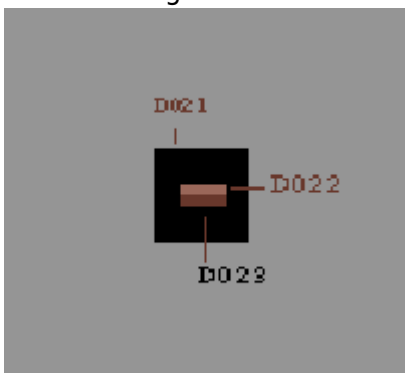
```
charset:    $2000-$2800
8 empty chars:  $27c0, $27c8, $27d0, $27d8, $27e0, $27e8, $27f0, $27f8
char values:    $f8, $f9, $fa, $fb, $fc, $fd, $fe, $ff
```

Now you need a positive and a negative image/char of your char bullet.

Negative image



Positive image



The negative has to be done with $d021 and $d800/colorRAM, the positive should be done with $d022 and/or $d023. $d800 should only be used for the positive when all chars got the same color RAM. Otherwise the bullets change their color accordingly everytime they move ahead which might look stupid.

Everytime the main program places a new bullet on screen respectively moving the char bullet on screen, it first has to read and save the char underneath the bullet.

```
ldy screencolumn    //column of char bullet
lda (screenposition),y  //screenposition=zp address holding the screen row
sta charstorage,x   //save char at specific position, x pointing at
```

```
currently used char bullet
jsr chargenerator   //now create new char and print it on screen
```

Now the new char has to be generated:

```
.const charsethibyte    = $20        /charset at $2000
.const negativechar     = $2008        //2. char in charset
.const positivechar     = $2010        //3. char in charset

//acc still holding value of saved char
//x-reg still pointing at currently used char bullet
//y-reg still holding current screen column

chargenerator:  asl                //figure out chardata of saved char
        asl
        asl
        sta tmp0        //get low byte of chardata in charset -> tmp0=zp
address
        lda (screenposition),y
        lsr
        lsr
        lsr
        lsr
        lsr
        clc
        adc #charsethibyte    //charset located at $2000
        sta tmp1        //get hi byte of chardata in charset
        lda charsetposition,x    //fetch low byte of new chardata to be
written
        sta writechar+1        //selfmod
        ldy #$07
!:      lda (tmp0),y        //read chardata from charset
genand:     and negativechar,y    //use negative char to punch a hole into
chardata
genora:     ora positivechar,y    //and fill it with positive chardata
writechar:  sta $2700,y        //write new chardata (low byte: selfmodifying
code, mind the hi byte)
            dey
        bpl !-
        lda newcharvalue,x    //get char value of new char bullet
        ldy screencolumn    //column of char bullet
        sta (screenposition),y    //print new char bullet on screen
        rts

charsetposition:    .byte        $c0, $c8, $d0, $d8, $e0, $e8, $f0, $f8
//low bytes
newcharvalue:       .byte        $f8, $f9, $fa, $fb, $fc, $fd, $fe, $ff
```

Check out these games to see how it looks:

http://csdb.dk/release/?id=16770 http://csdb.dk/release/?id=105361