Artfull Manual

Version 1.4 of of Manual created 9th/April/2023 by Z8AP for version 1.4 of the game

Description

This is a Sega Master System game like no other (to my knowledge). Upon starting the game you will see the six options below in the title screen animation. Pressing 1 or 2 selects the options.

Artfull is an original Sega Master System game. Two teams of human players, using designated artists, referred to as P1 and P2 in the game,take alternate turns as the team doing both the drawing as artist and as the the guessing team. Only the team of the current artist is allowed to guess the opposing team stays silent. Player's 2 of the red team with score represented by the red counters. Player's 1 of the blue team with score represented by the blue counters. The artists sends graphical commands (in the MergeDraw programming language) to draw the desired object (known only to the artist). The guessing team must guess the desired object to win a point. A clue of the type of object e.g. “creature” is shown at the bottom of image.If the guessing team is correct a point is awarded by a designated person(can be for example a non-player or the artist). One game consists of 4 points (i.e. each player must draw 2 images each). At the end of the game the team with most points wins. Artists either: draw over pre-selected images to express the associated object (or phrase) , draw an image entirely from scratch where the object to be drawn is noted on paper (referred to as “on your paper” in the game) beforehand. This is a human only game-in other words there is no vs CPU team-the (Master System) console is used like a “game board” (a bit like if 2 human players are playing computer generated chess board to use a random game example) but the important difference here is the console is essential to be used as the “board” ; unlike chess there isn’t an non-electronic version of this game because its based on programming on an electronic console . The number of blue counters and red counters shows the score for the blue team and the red team respectively in every game. The score is also shown below the image after every point played for. Though the game is based on programming it is designed to be fun to play so even non-programmers can program without maybe knowing! The aggregate scores over a series of games (drawing over existing images or draw from scratch) since the console is turned on is shown on the title screen see above. The numbers by the blue counter and red counter,on the title screen,shows the aggregate scores for the blue team and the red team respectively. The aggregate is over the sum of points for both types of drawing game modes. To reset the aggregate scores you need to turn the console off or on. The aggregate score may be used,for example , to compete which team gets the first to five points after two games.

At the moment the game is a Two-Mega Cartridge which means it has enough images/art works for three unique games (four images are required for one game) -which is enough to experience what the game is like to play for the drawing over art work mode. Having 15 images (which requires a two-Mega cartridge to store) does not have an impact on the mode where you draw all the images entirely. The code is designed so it can be expanded to accept up to 512 images (though it would have taken a long time -many days past the 27.3.23 deadline-to do all 512 images) and with 512 images it would be a Four-Mega Cartridge since each image takes 8K memory.
The game is coded to make good use of the Sega Master System Hardware in the following ways:
*It uses nearly all the graphical tiles to show the colour works of art and an easy to use well presented game GUI/MergeDraw programming GUI.
*It uses the fact that the Sega Master System is a console that does not have high resolution colour clash (not true for all Z80 based computers) this allows to draw over existing images as the player expects without colour clash artifacts.
*It uses the Sega Master System's built in feature to have a up to Four-Mega cartridge without any extra mapping hardware which works fast enough (accessing the memory past the 64K limit) to draw over the images in real time where needs be. The generous Four-Mega Limit (provided already by the console) means there is plenty of space not to use compression and actually it is required not to use compression (for the curious-I tried compressed and Z80 ASM decompressed images and that turned out to be too slow) . Instead the images are paged in (past the 64K limit) on the ROM and then drawn over (which is quicker than reading from loading the image into VRAM then accessing VRAM to draw over it). The game uses paging in slot 2 and two uncompressed full colour 8K images are stored in each 16K page-any commercial flash cartridge has slot 2 paging i.e. no custom mapper chip is required (not true for all z80 based system cartridge games).

Emulation

Tested and working with MEKA 0.80 Alpha.

Longer description with more technical details

Adding to the above description. To my knowledge there is not a high level language interpreter specifically created only for the Sega Master System. One of the main ideas for this game is to make a game where the game mechanic requires programming by a non-programmer player-in other words though the game uses a game mechanic based on programming it doesn't feel like programming to a non-programmer/player because it is so fun. This is done by using the MergeDraw programming language which is a programming language I specifically created and designed for this game and I created the MergeDraw interpreter from scratch in Z80 assembly language (it would be too slow in my opinion if the MergeDraw interpreter was done in anything else that isn't Z80 assembly language)-the language allows to merge your drawing, created by the MergeDraw program, with your chosen image, which is why it has the name MergeDraw. Because it is a programming language designed for non-programmers (I know that sounds contradictory-but I think you know what I mean!) it's designed to be fun and less complicated than maybe something like C++-after all we don't want the player to be frustrated by finding things like bugs in their MergeDraw program as that would not make for a fun game; though design to be less complicated than e.g. C++ the Z80 ASM code for the MergeDraw interpreter isn't straight forward as can be seen from the source code-some of the complexity comes for implementing MergeDraw's loop, in Z80 assembly, and the possible error messages for loops. The game features a full programming environment, entirely made from scratch in Z80 assembly, for the MergeDraw programming language. We will summarise here some interesting parts of how the source code/MergeDraw interpreter works. Each all the MergeDraw instructions,except PENUP, fits into exactly one byte: the first three most significant bits of the byte encode the instruction, the last 5 bits of the byte encode the operand value of the instruction. Since there are 8 bits there are 256 possibilities for MargeDraw instructions-but actually less than 200 are used for actual MergeDraw instructions. An empty space for the instruction is encoded as a value that isn't any instruction. A copy (in RAM) of the MergeDraw program is made with the new instruction entered,

this copy is checked for correctness for MergeDraw's syntax rules-a repeat loop takes a lot of syntax checking. To check for loop syntax the copy of the MergeDraw program is copied again (i.e. a copy of a copy) with all but the loop commands-i.e. the non loop commands are filtered out. This copy of a copy is then expressed in a canonical form which results in a sub-block of consecutive bytes, in the "filtered" copy of a copy,that have known patterns if there is an error of some type for a MergeDraw loop command. If there is an error in a MergeDrawloop the player is notified to correct it. If there are no syntax error's in the copy of the MergeDraw program the copy writes over the actual MergeDraw program that is actually used (in RAM) to update it. This all works because the Master System has enough built in RAM. For more technical details see the Z80 ASM source code, which has meaningful names. comments and description of how source code words (at start of source code).

Explanation of the Graphical User Interface in The Game

NOTE: THE BLUE OR ORANGE ARROWS OR RECTANGLES SHOWN BELOW ARE NOT PART OF THE GAME GRAPHICS

The user GUI is designed to be easy to understand. For every mode the control keys that are used, for that mode, are shown in the onscreen red or green boxes. All keys are on pad 1 unless otherwise stated. Pad 2 is used to delete or start over (i.e. it deletes the whole MergeDraw program and clears the canvas to the initial image)-this makes accidental deletion impossible unless the player handles pad 2. In game modes the pad is shared between player 1 and player 2 depending on if it is their turn to draw. The long red arrow that splits the screen into the "mainly the drawing part" and the MergeDraw program (on the right side of the arrow).

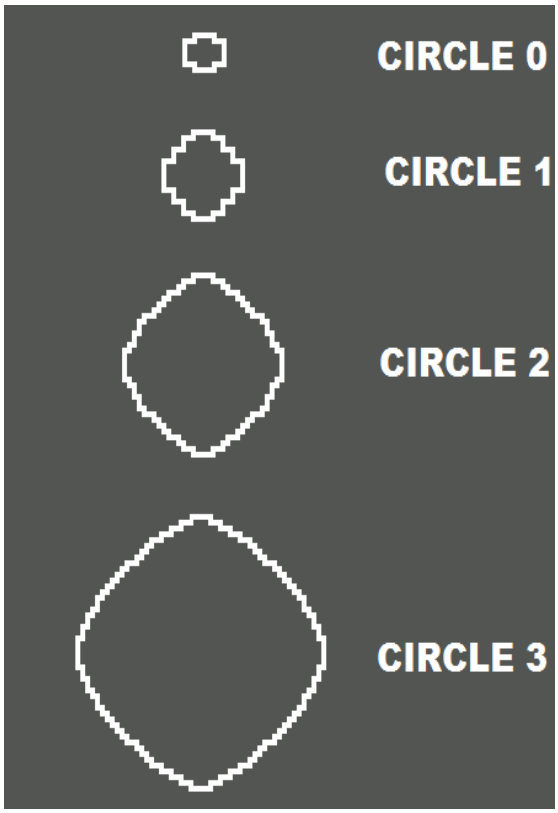
The MergeDraw Commands

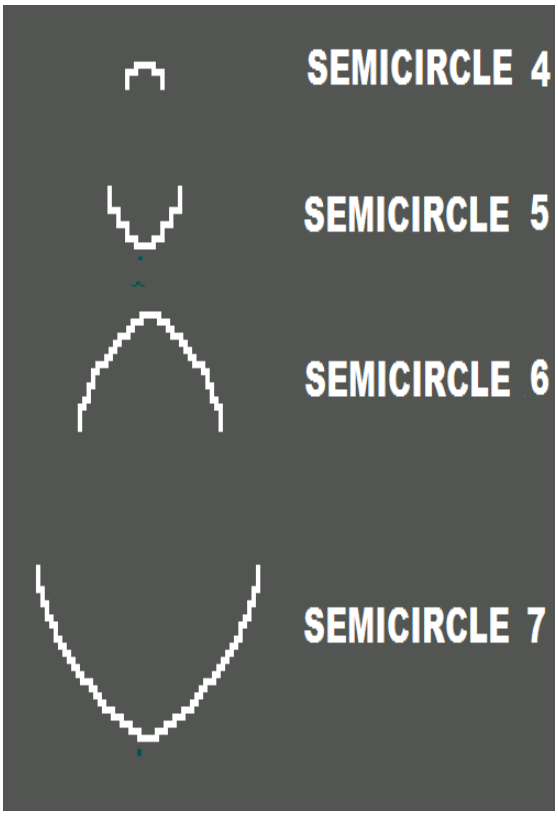
All commands that can be entered in the game are shown in bold below. One MergeDraw program can be at most 69 instructions long displayed over 3 pages; a page is shown on the right side of the long red arrow (that splits the screen) and does not take up the whole screen; 23 instructions are shown on a single page at any time pressing the right key for up to five seconds display then next page (one of the three pages). A command is selected by pressing 1 in game and the value for the chosen command is selected by pressing up or down. Less than 23 instructions maybe required depending on what needs to be draw in "GAME WITH YOU DRAW ALL" mode. The commands are described below but it is probably be figured out how they work without reading all of below, by some players. but below is given for completeness.
The nine commands are displayed in alphabetical order- pressing button 1 goes forwards in the menu or pressing the left key goes backwards in the menu.
There are exactly 9 different commands in MergeDraw which controls the hedgehog cursor that draw on screen using a virtual pen (so the pen position is given by the hedgehog).; the hedgehog cursor may wrap around to the opposite edges of the canvas and draw if needs be-i.e. it does not stop a screen edge if it runs out of space to draw. The 9 commands in MergeDraw are as follows:

- The CIRCLE command

CIRCLE 0-the smallest diameter circle
CIRCLE 1-a medium diameter circle
CIRCLE 2-a large diameter circle
CIRCLE 3-largest diameter circle
SEMICIRCLE 4-the smallest semicircle top half
SEMICIRCLE 5-a medium semicircle bottom half
SEMICIRCLE 6-a large semicircle top half
SEMICIRCLE 7-largest semicircle bottom half

The shapes and relative sizes of the above circle and semi circle commands are shown below.





The diameters of the circle or semicircle are fixed and cannot be changed (when I was creating the MergeDraw I thought that the user specifying specific diameters make it feel too much like a “non-gaming programming language” and this not compatible with making MergeDraw fun to use- also specifying diameters its not required and all the commands are flexible enough to express most things-or to get the idea across in the drawing-I think).
Note that the semicircle command may be selected by firstly choosing the circle command then using the up keys so that the operand is 4,5,6,7 is chosen. Similarly the circle command may be selected by firstly choosing the semicircle command then using the up keys so that the operand is 0,1,2,3 is chosen. The semicircle and circle commands are grouped together like this as they are obviously related. Or alternatively the SEMCIRCLE or CIRCLE commands can be selected from the alphabetically ordered menu using button 1 or the left key.

Note the CIRCLE or SEMICIRCLE command can only be drawn after a PENDOWN command (its because using a CIRCLE command after PENUP doesn’t really add anything -moving the pen can be equivalently done by straight line commands- and make the MergeDraw program more complicated than it needs to be for the player- MergeDraw is about simplicity). If you try and use CIRCLE or SEMICIRCLE after PENUP the relevant error message appears and prevent you doing so.

Another thing to note about the SEMICIRCLE command is that the pen position is not changed when used and finished drawing the semi-circle and in this connection the circle command is different. Because with a circle command the pen position is changed to the bottom of the last circle drawn-i.e. the top of the circle is drawn from the current pen position and the final pen position is downwards in the length of the diameter of the circle drawn from the previous position. Having the circles and semicircle behave in this way, regarding starting and finishing endpoints, and their 8 different shapes and sizes I think makes a good variety to draw most things,requiring curves,easily and MergeDraw is about being easy to use while being expressive enough to convey the words to be guessed.

- The UP Z command where Z is a number from 2 to 31 units to be drawn-draws upwards Z units from the current pen position;*where each unit is 2 pixels*. **UP 2,UP 3,UP 4,…,UP 31** are valid drawing upwards commands.

● The DOWN Z command where Z is a number from 2 to 31 specific units to be drawn-draws down Z units. **DOWN 2,DOWN 3,DOWN 4,…, DOWN 31** are valid drawing downwards commands.

● The LEFT Z command where Z is a number from 2 to 31 specific units to be drawn-draws leftwards Z units. **LEFT 2,LEFT 3,LEFT 4,…,LEFT 31** are valid drawing leftwards commands.

● The RIGHT Z command where Z is a number from 2 to 31 specific units to be drawn-draws rightwards Z units. **RIGHT 2,RIGHT 3,RIGHT 4,…, RIGHT 31** are valid drawing rightwards commands.

● The **PENUP** commands lift the virtual pen from the canvas so it can be moved without drawing using the UP, DOWN, LEFT or RIGHT commands. The colour for the pen ink is chosen from the menu before the game start-it can either be a plain colour or a multi-colour; each of the two ink types suits different types of images better.

● The **PENDOWN** command is used after a PENUP command to enable drawing again.

● The REPEAT Y command is always used in conjunction with an END REPEAT command, and REPEAT Y command repeats the block of MergeDraw commands it encloses Y number of times. The above two commands are MergeDraw's loop command.A REPEAT command cannot contain another REPEAT command-in other words nested loops are not part of MergeDraw-one reason is when I was creating the MergeDraw programming language I thought that using nested loops makes MergeDraw too much like a 'non-gaming programming language' and would make things less fun (e.g. making bug finding more difficult for the player). Also it's not required all the commands are flexible enough to express most things I think.
REPEAT 2, REPEAT 3, REPEAT 4,…, REPEAT 31 are the valid loop beginnings with the end of the loop finishing with **END REPEAT**. These two commands are always used in pairs.

MergeDraw Error Messages and their Meanings

The following error message in red are maybe display for some error,; their meanings are also given.

PENDOWN FOR CIRCLE When this is shown it means you are attempting to draw a circle while the pen is up. So use the PENDOWN command firstly to continue to draw the circle.

PENDOWN FOR SEMICIRCLE When this is shown it means you are attempting to draw a semicircle while the pen is up. So use the PENDOWN command firstly to continue to draw the semicircle.

NESTING ERROR When this is shown it means that you have attempted to put a repeat loop inside an existing repeat loop.

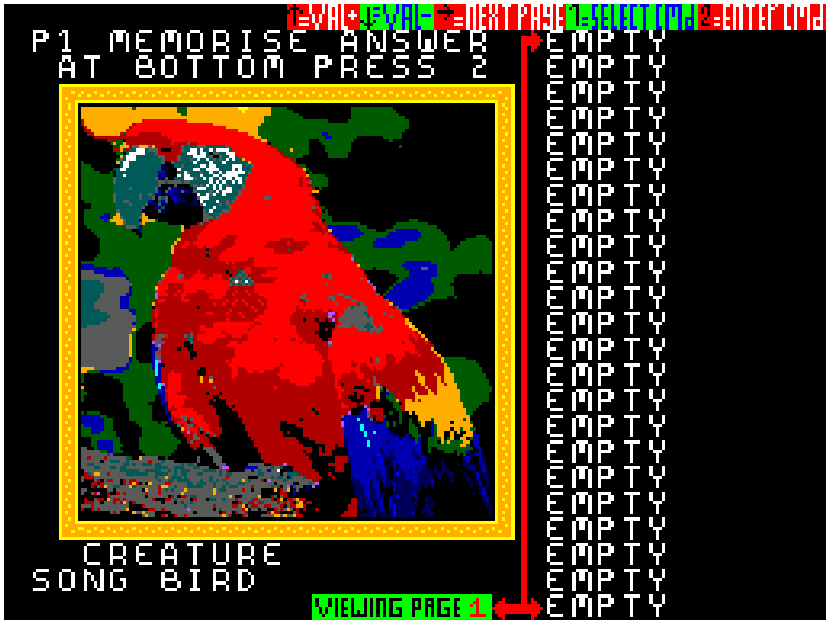
NO START ENDREPEAT When this is shown it means you you are attempting to start a loop using the ENDREPEAT command. It is also shown if the an ENDREPEAT is attempted to be entered as the last (69^{th}) command since it is pointless to attempt to start a loop on the last command.

USELESS ENDREPEAT When this is shown it means you have you are attempting to put an extra ENDREPEAT in the MergeDraw program; a loop only requires one ENDREPEAT for expressing where it ends.

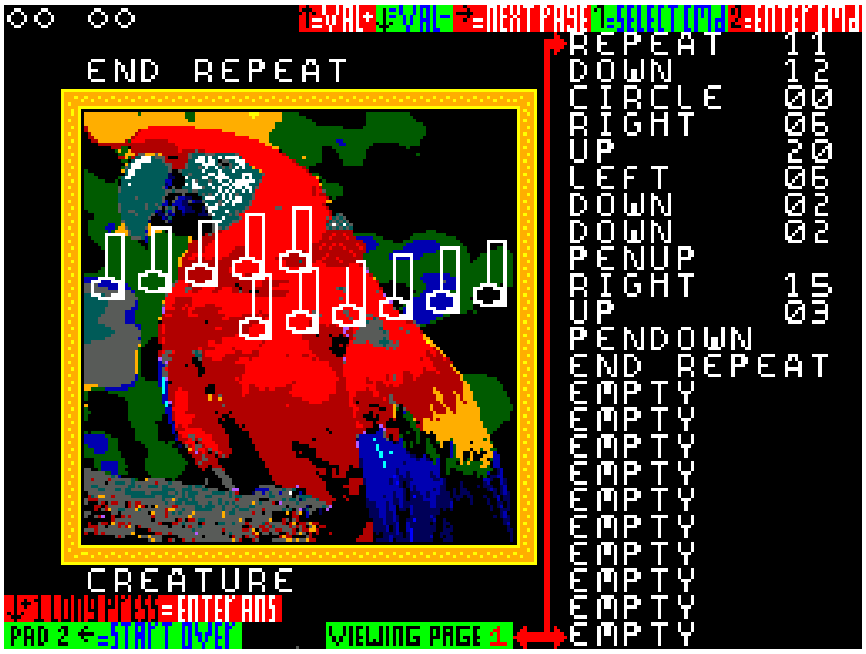
CANT USE REPEAT NOW When this is shown it means you have tried to start a loop at the 69^{th} command-which would be pointless as it cannot be ended with an ENDREPEAT command.

Three MergeDraw Program Examples

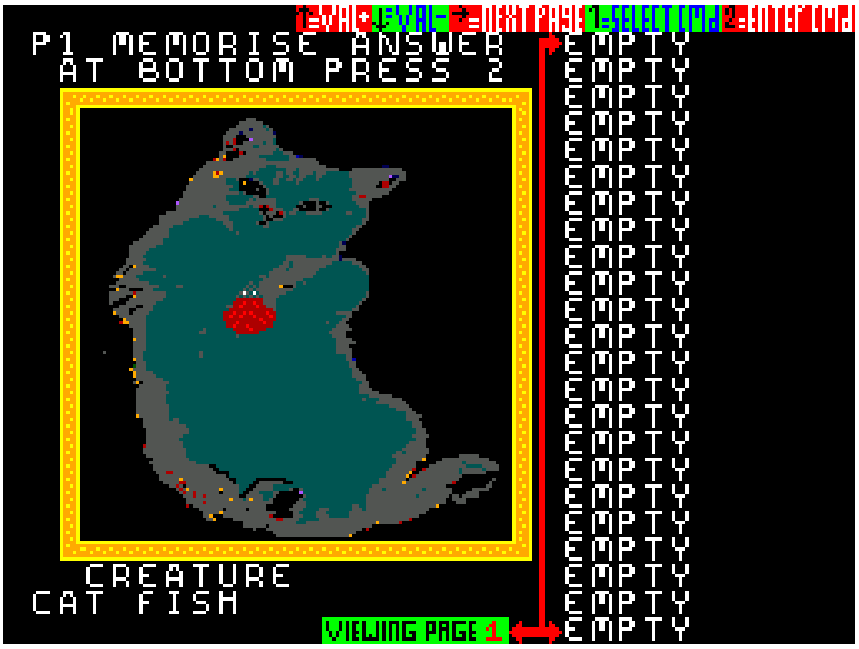
In this first example the image to be initially draw over is shown below along with what the thing that needs to be draw (a “SONG BIRD”) so it can be guessed by artist’s team-at this time the artist’s team is looking away at the screen so doesn’t see the answer; but the opposing team can see the answer (but stay silent so as to not give away a possible clue to what the answer is).



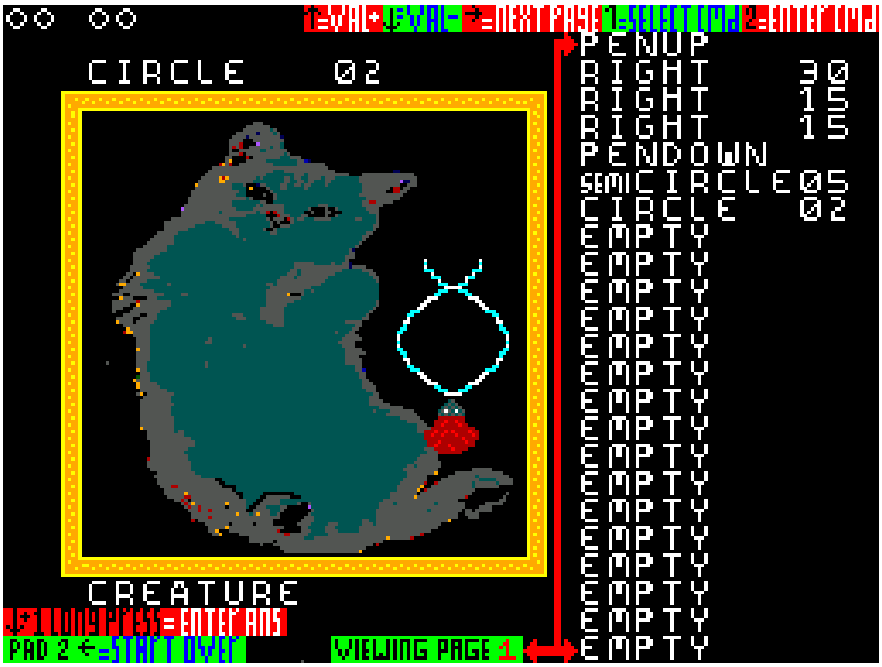
Below shows one possible MergeDraw program to express the creature “SONG BIRD” by drawing musical notes over the image.It needs just 12 instructions-less than ½ the first page shown below. This example uses a REPEAT 11 loop command to draw the eleven notes.



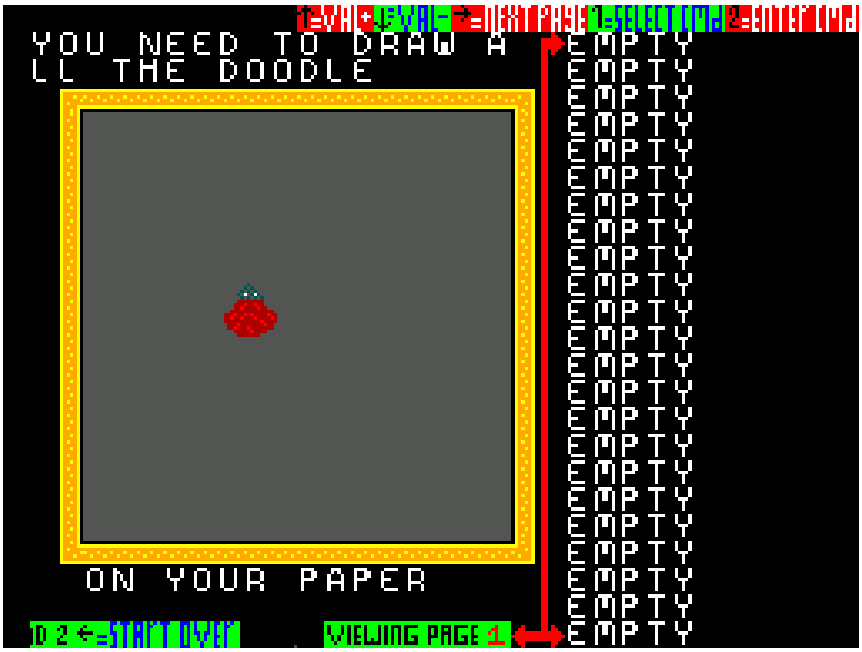
A second example. The image to be initially draw over is shown below along with what the thing (an aquatic creature) that needs to be represented by drawing (a “CATFISH”) so it can be guessed by the artist’s team-again at this time artist’s is looking away at the screen so doesn’t see the answer but the opposing team can see the answer.



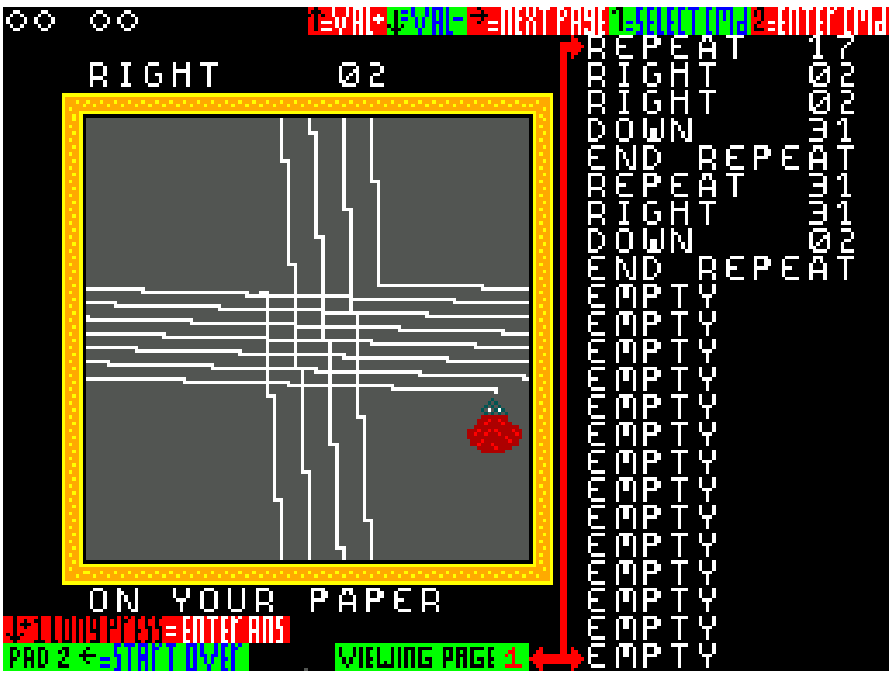
Below shows one possible MergeDraw program to express the creature ‘CATFISH” by drawing a representation of a fish over/beside the image. It needs just 7 instructions-less than ½ the first page shown below.



A *third example* is in the “GAME WITH YOU DRAW ALL” mode where the player starts with a blank screen as shown below.



The teams has decided for their two images, what to draw (i.e. what is to be guessed by the other player) before the game starts and wrote it down on paper-which is what “ON YOUR PAPER” in the above image is referring to. During the game it is not allowed to make any changes to what has been expressed on paper. In this example the player of the team noted down the object “crossroads” on paper. The teams may optionally choose to note down on paper a clue of the type of object for the guessing team to see. Below shows one possible MergeDraw program to express the object “crossroads”. It needs just 9 instructions and uses two loops-less than ½ the first page shown below. It can be confirmed using what was written before on paper if the guessing team is correct or not.



Control Keys

The control keys are displayed on screen for the chosen mode and also described below. Pressing 1 on the second control pad undoes the last command in any mode (this undo feature is actually a cheat though the cheat being activated through the menu is not yet implemented so it so simply works by pressing the button 1 on pad two at the moment). All commands can be cleared by pressing left on the second pad.

Control Keys in Any Mode i.e. the Two Game Modes and Programming Mode

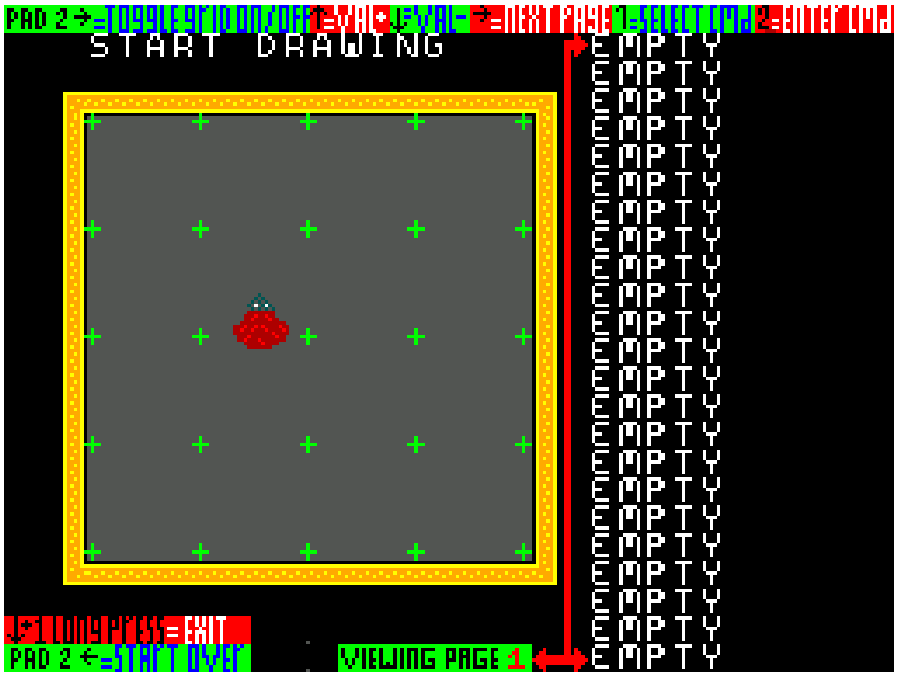
Pressing 1 or left selects a (MergeDraw) the chosen command and pressing up/down immediately after selects the value for the chosen command; pressing 2 enter the chosen command with the chosen value. Pressing right changes the page of the MergeDraw program listing being viewed.
In any of the three possible modes the game uses both control pads. Pressing left key on the second game pad, in any mode, deletes (or starts over) the drawing and the MergeDraw program listing if a mistake is made. The game is designed so that individual MergeDraw commands cannot be deleted as I think that makes the game too easy. The second control paddle is only used to delete a drawing and every other control key, in any game mode, is with paddle 1. Mode specific control keys are described below.

Control Keys in Any of the Two Game Modes

In any of the two game mode (that is chosen by either the “GAME WITH DRAW OVER IMAGES” or “GAME WITH YOU DRAW ALL” menu options) the keys are as follows. Pressing down key+button1 together for up to 5 seconds in either game mode enters the answer (it displays ‘ENTERING ANSWER’ until the answer is entered).

Control Keys in Programming Mode

In programming mode (that is the chosen by the “MERGEDRAW PROGRAMMING LANGUAGE” menu option) pressing down key+button1 exits back to the menu. Pressing right on the second control paddle toggles the grid on or off. The second control paddle is only used to delete a drawing or toggle the grid on/ off (grid turn on is shown below); every other control key in programming mode is with paddle 1.



Game Credits

WWW.UNSPLASH.COM

for public domain images.

All other graphics by Z8AP.

Music by Naomi Z. Nosipho.

All code created in 100 percent Z80 assembly language using VASM assembler entirely by Z8AP.