MJSW v3

Scope

This combination of Manic Miner and Jet Set Willy, has undergone a progression from V1 through to V3.

Shortening of names used throughout this text.
The full name of Jet Set Willy will be shortened to JSW.
The full name of Manic Miner will be shortend to MM.
The full name of Manic Jet Set Willy will be shortened to MJSW.

V1 version of MJSW was the compression of the two games into 48k. Which allowed both games to be played separately. Or if skilled enough one after the other.

V2 version of MJSW removed the game independence of each other and integrated the games. Now The main game was JSW and the option to visit MM caverns was spread around the Mansion. Finishing the JSW part and the MM part was the game completion.

V3 version of MJSW is not a great deal different to V2. The game itself is more or less identical. Here the focus of my attention was adding more seamless transitions throughout the demo and other title graphics.

Purpose

This started out as just an exercise in compacting JSW and MM. Which is a far bigger a challenge than just editing the rooms and data. In ver-

sion 1 (V1) the basics of the two games was compacted and run. Not exactly, what I wanted. By version 3, the programme is more in keeping with the original plan. In v3 you can play all the rooms of MM and all the rooms of JSW, the changes I made are fixes needed to either remove the original game bugs in JSW or additions to make the game play better. I have not changed the original games much, leaving the level of play a very close match to the originals Increasing difficulty was not my aim.

The keys to play will be a close match to the original games

When the music on a title screen is playing or when the message is scrolling past.

<Enter>	start the game
<1>	(numeric one) change the Portal play

When the game is playing

<QETUO>	moves Willy left
<WRYIP>	moves Willy right
<ZXCVBNM>	make Willy jump
<Shift>	make Willy jump
<ASDFG>	pause the game
<J>	speed the game up
<H>	slow the game down

This text is not definitive and more than likely contains errors. The whole lot is from recollection, without bothering to check and verify specific numeric data.

Differences

In a normal JSW game the clock counts the amount of time Willy has been active since the game start. In MJSW this is the same. However time is now costing Willy monies for the upkeep of his house. So every minute a small sum is taken from the displayed ready cash. The ready cash can be supplemented by collecting glasses in the JSW mansion/ house. The sum rewarded for such collection is just a token value. A far greater value is attached to the items that are accessed through the Portals spread around the house. In the MM caverns the items are worth a lot of monies. These are the same items that Willy originally collected to make his fortune (hence their increased value). On completion of a portal accessed MM cavern more monies will be awarded for the remaining air.

Whilst time slowly takes monies and items and finishing MM caverns gives monies. Their are also far bigger drains on Willie's ready cash. The cost of a heavy fall or touching a nasty or a guardian will incur a heavy financial cost.

Specific MM style caverns also add either a bonus or a penalty
In the solar power room, Willy can suffer severe burning which incurs a financial cost.
The Kong rooms, award monies for dropping Kong.

Completion of any MM cavern will remove further access to the cavern from the appropriate access point in the JSW room.

The extra lives given for score in the MM caverns is not carried over into MJSW.

The changes from normal. JSW and MM.

Most of the code and data changes will not be noticeable. So for most of the game it will appear as if no change has occurred. However what you

see and what has been changed is very different matter.

Here I will list the visual differences, that can be seen.

- Large graphic - stating “stop tape and press enter”. Here we have a full sized screen graphic. That is an addition to the original. This graphic has a colour changing diamond graphic.
- Credits screen with graphics
- The MM title screen with revised, with corrected piano graphics. Animated sun reflection on the water, and full key piano illumination when playing the tune. A new graphic stating “MANIC JET SET WILLY”.
- double height scrolling message. scrolling at a rate of 4 pixels at a time.
- The JSW style title screen has a good deal of graphic changes. But each change should leave the overall design looking very similar.
- JSW title has 3d shadow
- JSW title has pixel outline
- JSW title has new colour sequences for the Penrose triangle
- JSW title has new colour sequences for the large text.
- JSW Title -- the colour sequences are not set in a similar method to the original colour cycling used in JSW. The code actually works out a new set of colours which has no direct chronological order. This ensures the JSW title screen has literally hundreds of colour graphic combinations.
- Screen displays that are not shown outside of game play. Are brought to the screen using one of a multitude of pull on curtain routines.
- The curtain routines are in essence random in what method is used to display a screen.
- The order of rooms displayed in demo mode is constantly changing
- The insertion of Title screens interspersed between the game playing screens is essentially random.
- The demo loop displays the words “attract mode” which is being constantly colour cycled.

Now the playing game.

- I added into the basic JSW playing area, the missing four screens. So this version has 64 playing screens in JSW. Whilst I added the screens, the added screens are intended to tidy up the game layout. And all the added screens are not such played.
- Maria has more interaction with Willy
- The game completed hop on the bed. Is changed to exactly that. To activate the game ending Willy must jump onto his bed.
- Some (not many) sprites are reversed. Why is not important here. Code change allowed the reversal, so in a few cases I reversed the sprite.
- The nightmare room also has interactive sprites.
- The nightmare room has one new sprite.
- Scattered around Willie's mansion are portals
- The collection of JSW objects now counts down towards 0, which is the completion of the JSW part of the game
- The number of caverns completed in the MM part of the game counts upward. The count of caverns completed is displayed to the right of the <MM> text on the status line. When it reaches 20 and all the JSW items have been collected. Then the game has been fully completed.
- The collection of items in MM caverns will increase the number of MM items collected, The total displayed is on the left of the <MM> text displayed on the status line.
- The monetary value of each item collected varies between those that must be collected as part of the house tidy up, and those that are scattered around the MM caverns.
- When all items in a MM cavern are collected and the bonus has been awarded for the the stepping into the MM portal. The portal that gave access in the JSW mansion, will be deactivated.
- The tune now plays a big role in changing the games speed. Pressing the keys "H" and "J" whilst playing the game. Allows for a change in the music speed. A graphic is displayed to give feed back on the selected speed. When the speed is set to the slowest. The graphic dis-

plays the word “SPEED”. When set at its fastest just three dots remain of the “SPEED” graphic. The default speed is with just the letter “S” showing.

- Changing the game tune to on or off via pressing the “ENTER” key. Will have little change in the overall game speed. This is because even when the music is switched off . The music routine is active and setting the speed of play.
- An icon is displayed in the status area indicating when the music is playing. This is supplementary to the dancing Willies. (which by the way, can not show the music status. When their are no Willies displayed)
- Each new room/cavern visited, is added to the rooms visited counter displayed in the status area. That count increases for both JSW and MM room/caverns
- Pressing pause will not only pause the game. keys <ASDFG> . But will also display the current high score. The high score will be displayed underneath the present score.
- The colour cycling of pause is deliberately slowed down, to enable easy and rapid reading of the high score. It also allows a brief window to look at a screen without the constant colour cycling of old versions.
- The high score is constantly being updated, and is not taken from the final end game score. So even playing the game once will generate a high score entry. Just from the score at the game start.

Portals

- When the title screen is displayed, along the bottom line will be text that is stating the method portals are used and accessed in the game.
- Pressing “1” on the title screens when the music is playing or the scrolling message is being displayed. Will change the method of access from the portals. There are 4 modes, which can be treated as only 3.

1. “fixed by play”. Here the access to each MM cavern is ordered in the same order that they are accessed in MM. No matter which room is

used to access a MM cavern the order is fixed.

2. “fixed by room”. Here the access to each MM cavern is defined by the room the portal is in. So specific JSW rooms give access to specific MM caverns. The caverns can be played in the order that the portals in the JSW rooms allow access.
3. “random by play” and “random by room” Whilst this is technically two differing modes of play. I will describe them and treat them as just one mode. When either of these modes is selected, the room access or the order access will result in an unknown MM cavern. For every new game played. The order will be unknown.

- In order to help the player to decide, on using the entry to a MM cavern via a portal access door. The graphic displayed on the portal will match the MM caverns portal graphic. If however playing in any of the random modes of play. The graphic displayed on the JSW portal access door will simply be a question mark.
- After completion of any MM cavern, the access portal that was displayed in the JSW part of the game. Will be removed.
- Any MM cavern can only be played once per game cycle. In order to return to the JSW room that gave access to the MM cavern. The MM cavern must be played through to completion, with the cavern bonus given for the remaining air.

Game changes.

- After each contact with a deadly object/sprite or heavy fall. Willy is repaired at a cost to himself. The long line of Willies along the bottom, is an indication of how many body repairs Willy is allowed before the damage is considered too extensive. On the loss of his final body repair the game is over.
- After a body repair a brief period of immunity is allowed. This period of immunity should remove all the old infinite death loops from the game. So if Willy wants to, he can do the jump from high up in the house over the flag pole and down, down, down to his demise, At the cost of only one repair. Unlike in JSW2 and JSW2+ he will be left at

the point of the jumps landing.

- Note that body repairs cost a lot of monies. The initial supply of ready cash will not cover all the repairs allowed. If for whatever reason his monies deplete to nothing. Then that is also game over.
- Collecting the objects in the JSW rooms will only give a token repayment to Willie's cash. The bigger value items are in the MM caverns.
- The passage of time is indicated on the game clock. For each change of the game clock, monies will be taken from the ready cash that Willy has. The amount taken is needed to pay for the staff and the running costs of his house.
- The solar power room, has my normal colour changing beam. The beams graphic is extended into the point of contact in the room. The point of contact will be illuminated in various colours. The beam hitting Willy will turn red at the point of contact. The beam will also generate a sound, and sap air from Willie's air supply. In addition the beam will add damage to Willy and this damage must be paid for. So in addition to the air being taken a large amount of cash will also be taken to cover for the cost of skin damage. If daring to play with random game order. Accessing the solar power room without much cash can prove to be rapidly, a game over situation.
- MM in game tune, plays for the MM caverns
- JSW game tune plays for the JSW rooms.
- The graphics depicting the number of body repairs that Willy is allowed. Will match the graphics of Willy playing the cavern or room. E.g. the hat is either a helmet or a top hat

The Game changes.

I left the parts that influence how the game is played. This meant I actually had to edit the known flaws and leave them as they were. I normally remove the features as written into the original, which strictly speaking should not be there. Simple flaws that allow Willy to pass his head into wall blocks moving left. Jump diagonally into floor blocks and jumping

into wall cut outs going left that can not be performed going right.

Some of the flaws can be removed by changing the game structure. So simple things like flickering banner text, when moving from room to room. Have been removed without impacting on how the game plays. Other changes that remove the spites flickering and also Willie's flickering (which I have named the jagged finger look - and is described by me in other documents) have been altered as well.

I would expect playing this version to look and feel quite close to the original.

Behind the scenes. -- This starts to get technical.

Ignoring the data changes that occur in modified versions of JSW or MM. Here I am talking of code changes. Slight modifications to either JSW or MM can be achieved very easily in the standard 48k games because both games have unused space in them. The vast majority of modifications to JSW and MM use the already free space to add enhancements or modifications. These changes are to me best described as trivial game changes. In most cases the total change is less than an hours work for me. These types of trival changes are,

1. swapping the LDIR routine for LDI
2. swapping the LDIR routine for a stack copy
3. Adding an extra tune.
4. playing in Mono colours.

Note I am taking about code changes and NOT the changes done by editing the graphics/sprites/tiles/room plan/sprite paths. These types of changes are just changes in the data. For more extensive game code changes the normal path is to move to a machine that did NOT exist when the originals of JSW or MM were written. Namely the versions of the 128k spectrums. To write MJSW on the 128k version of the spectrum is not exactly a hard task. just store the original games in their entirety,

and swap between them as wanted. To write both in 48k and run either JSW or MM at will, takes a lot more effort.

The size of the basic game in both JSW and MM is reflected in the size of the game files used to create the original games. In both cases the game file size is around 32k. Analysing the buffers and screen size from both JSW and MM, we find that these buffers are allocated just short of 16k. Which when then looking at the memory of the 48k spectrum indicates around 32k is the size allowed for the game code and its data. Which is a match for the file that is loaded. Matthew did not do file memory reallocation. Which explains the near enough memory match.

The data loaded does also includes the unused data parts that can be used for code enhancements. In JSW we can overwrite the entry code part and also the 4 empty rooms at the top of memory. In MM the part unused is a small area below the room data. (+ some unused code in JSW)

The reason these parts of unused code are included in the loaded game files is simply due to the method Matthew used to transfer memory from the Tandy to the Spectrum. (not elaborated on here)

The two games need to be squashed to still fit into the same allocated space of around 32k, if the dual game is desired. This means a reduction from 64k down to 32k for the combined code and data from both games. As mentioned above some of that data can be immediately discarded. But the vast majority is still needed.

The data that needs to be squashed/compressed consists of many parts. The overall aim is a 50% reduction of each and every part of the data. The code part will be rewritten to run both games. That code will also need to be reduced, even with the extended usage, down to the original size. First we look at the data.

The data part can be split into the types of data.

Data parts.
1. room plan
2. room tiles
3. Sprite definitions
4. title screens
5. In game music
6. Title screen music
7. Ropes
8. Item positions
9. Sprite paths
10. Lower screen colours
11. The Y table

The various parts underwent changes between MM and JSW. The most significant was the change in how the rooms were stored and how the room plan was laid out.

DATA part 1) the room plan

In MM the caverns plan is described by a continuous stream of 512 bytes. Each byte is used to describe a graphic tile. And in this manner we can describe all the platforms and walls in a cavern. For 20 caverns we allocate 20*512 bytes or 10240 bytes for 20 caverns.

In JSW the room count has tripled compared to MM. Allocating a stream of 512 bytes for the room plan is not a realistic proposition. So Matthew changed the method of storing the room plan. We now have two bits describing each tile, instead of a byte for each graphic tile. Two bits allows the scope of defining 1)Empty, 2) floor, 3)wall, 4) nasty. Which is then overlaid by one set of stairs, and one animated conveyor. So the room plan changes from a stream of 512 bytes to a stream of 512 pairs of bits. Which can also be described as a stream of 128 bytes. This redesignation of the room plan has substantially reduced the space needed in JSW for the 64 rooms that this game has. So JSW now needs 64*128 bytes to

store its room plan.

The reduction from 512 in MM down to 128 in JSW is a good saving. It is achieved by also reducing the amount of tiles used per room. MM can have two types of Nasty tile JSW can only have one. MM has a tile designated for switches, and a crumbly tile as well. These tiles are not included in JSW style rooms.

In JSW2 the room plan byte count was further reduced to around 60 or 70 - I am not looking up the specific numbers. The reduction was implemented by using a few lines of Basic on an Amstrad CPC 464. The basic simply counted how many tiles were similar in a data stream and changed the data stream to be a combination of count and tile type.

In JSW2+ the tile count was reduced further by duplicating the stream to scan the room horizontally/vertically and have an odd tile counter. The net result was a drop to around say 50 or 60 bytes per room. Whilst this figure does not sound a lot. these are average byte counts per room. The saving of just ten bytes per room must be paired to the room count. JSW2+ having 147 rooms, does indicate that the method saved another 1470 bytes.

MJSW V1 used a method similar to JSW2. In this case the room compression was written in QB64.

MJSW V2 uses a newer method. The data is scanned (in QB64) to find horizontal platforms. If found it then scans below the platform to see if the platform has a height. It keeps scanning downward for the height. It then stores the width height, type and position. Then deletes the platform from the room. And scans the room again. Doing the above will slowly delete all the graphics in a room. When the room is empty. The routine outputs a stream of data describing what it has found. The data stream changes the number of bits output, depending on what is being described in the stream. The net result of the above is the average size of each room is reduced down to around 40+ bytes. What it also does, is de-

scribe both MM and JSW rooms in the same style data stream. Even though The MM data stream has a greater bandwidth. This reduces the the data for JSW and MM rooms combined to less than either of the original data sizes.

Data part 2) the room tiles

In JSW and MM the tiles are stored in the rooms data. In Manic Miner the number of tiles per room/cavern is greater than the similar JSW room. In MM we have

1. Background tile
2. Floor tile
3. Wall tile
4. nasty1tile
5. crumble tile
6. Nasty 2 tile
7. switch tile
8. conveyor tile
9. item

so for 20 caverns we have defined 180 tiles

In JSW we have

1. Background tile
2. Floor tile
3. Wall tile
4. nasty1tile
5. Ramp tile
6. conveyor tile
7. item

So for the 64 rooms of JSW we have 64*7 tiles or 448 tiles.

In order to combine the two games this data needs to be reduced. The room plan was slowly over various versions reduced from the MM size of 512 bytes to the final MJSW V2 size of 40+ bytes. It needed a similar reduction for the tiles.

I wrote another programme in QB64 that scanned the tiles and looked for duplicate tile definitions. Whilst at the same time changed the tile definitions stored in a room to tile indexes. A similar method as used in JSW2. Whilst in JSW2 the index system was done by hours of scanning the data and simply redrawing all the tiles. Here I set the basic programme I had written to delete the tiles as needed without looking at the data myself. The basic indicated that I could delete from the 448 JSW defined tiles around 180 tiles. In a similar way I could delete from the 180 MM tiles around 120+ . Leaving me with around 320 tiles in total. This compaction has reduced the total tile count for both games to less than the original JSW. But still more than the original MM

Data part 3) sprite definitions.

The sprite definitions are not as compliant in compression as the previous two sets of data mentioned. In actual fact the sprite definition do not readily compress at all. The data does have several places where the data is null. The problem is there is no real gains to writing code for those few instances.

I wrote some code in the summer 2022 to just test the speed difference in having the sprite definitions reversed by code alone and not have a reverse definition defined. After a quick test I found the code I wrote had no great speed change when implemented. So I undertook a quick change of a stock JSW game file. Then moved onto the the latest version of MJSW v2 - No idea what subversion it was. But it only took a few hours to incorporate the change into the file. The oddities, quirks and exceptions took longer.

In order to have the sprites change definition by code, needed a

good deal of extra code. The amount of code written is nowhere near the amount of data that is deleted. So in MJSW V3 a lot of sprite data space is re-used for other game code or data.

Data part 4) title screens

The aim again is to halve the needed space, in order to fit into memory both of the title screens.

In MJSW v1 I managed to only display the JSW styled title screen. Even though I still had stored the top part of the MM title screen I simply did not have enough memory to create the full MM title screen.

Matthews original title screen needed to store 512 bytes of data in order to draw the Penrose triangle. My code and graphic redesign reduced the size down to somewhere around 80 bytes. A good saving but in V1 I still did not have the space for the Full MM title screen.

IN V2 The memory saved by the new room plan compression was immediately used and the rest of the MM screen was created. Unlike the way the screen was created in the original MM, my screen is built using code and data. The original screen was 4.5k of data. My screen is as stated built from various pieces of code.

The top graphic picture, which is also used in the Final Barrier is compressed by a control byte system. The rest of the screen is built using various routines. The piano graphic uses around 10 bytes of data. Yes I did write 10 bytes of data - the piano in Matthews uncompressed data uses 512 bytes. The 10 bytes of data does need a routine to expand out the data into the original 512. The expansion routine is 45 bytes - so still a big saving. The expansion routine draws most of the repeat graphics that exist on the middle part of the screen. Leaving only the large MANIC JET SET WILLY, which is compressed and drawn using the same routine as used for the top picture. Overall the MM title picture

routines and data usage is a lot smaller than the Original MM of 4.5K - combined the new JSW and MM title screens are comparable in size to the original MM title screen.

In V3 the JSW title screen undergoes a lot of new graphic routines. I wanted to write these routines and I will admit that the amount of code used does not really justify the routines inclusion. The graphic changes on the JSW title was really the only big change I could do, that did not affect the game play.

Data part 5) in game music

Not much scope to compress- in any case the size of the data is so small, that attacking the data with a compression routine does not leave much to be gained. So it was left alone.

Data part 6) Title music

The data for the MM title music was an old challenge for me. I have listed somewhere a method of compressing the MM music data. The data itself is compressed to a third of its original size. But then needs around 20 to 30 bytes to enable expansion back to the original

The data for the JSW title tune was left. Just a slight redesign of the playing routine.

Data part 7) Ropes.

This data is specific to JSW and Matthew allocated 256 bytes for the data. Another problem that I have written about and I have listed a method of reducing the data from 256 bytes down to around 42 bytes.

Data part 8) item positions

This data undergoes a major change between MM and JSW. In MM the data is stored inside the room data. In JSW it is stored in an item list. This does present two problems. The first problem is the reduction of data - the second problem is the data usage and control. Which I will ignore here. The item data for JSW is assigned 512 bytes of memory for all the rooms, the similar data in MM is 25 bytes per cavern, or 500 bytes. Since I am combining game data, I decided to use a similar method for MM data as used in JSW. Note I said similar and not the same. Whilst Matthew laid out the data with low nibbles and High nibbles of data spread over two pages of data. My data is kept together, is not page aligned and is not searched in the game loop. The result is the combined data allocated for both JSW and MM is around 380 bytes. So another combined reduction of data.

data part 9) sprite paths.

The paths of each sprite is stored in the room space for each of the 20 MM caverns. For the 8 sprites paths defined per room Matthew Allocated 56 bytes, for the 20 caverns this is 1120 bytes. For JSW the method changed, the room space describes a guardian to use and then that guardian is placed (slightly modified) into the playing room. so JSW has two sets of data that is used. For the 64 rooms we have a possible eight guardians, with each guardian defined as two bytes. So 64*8*2 bytes or 1024 Onto that needs to be added the 127 ish guardian definitions of 8 bytes each. so under 1024 bytes - combined around 2048 bytes. For MJSW I decided to use a similar style to Matthews method In JSW. The game data was split into JSW playing data and MM playing data. I will just give overall data sizes for MJSW (which is the combined jsw and MM data). The overall number of guardian paths defined in MJSW is around 200, but these guardian definitions are only 7 bytes so memory allocated is 1400. The room allocation for these guardians is ?? , which

still use two byes each or around 500 bytes. which combined gives under 2000 bytes. So again the combined data total is less than just the original JSW data total. The reduction is mainly due to not allocating data for sprites that do not exist. In MM and JSW the sprite allocation is fixed. My data varies and only defines used sprites.

data part 10) lower screen colours

Another wasteful piece of data in the originals of JSW and MM. The lower 256 bytes of screen are allocated a byte each to define the colours. I have tended not to use data in this way. In my case I create two lists of data. The 1st contains all the colours that exist in the lower screen. The 2nd list is the repeat count and the index to the first set of data. The typical data reduction is from 256 down to 50 bytes. The code is used in multiple instances and is used to colour multiple screen areas. It is also the basic code need to draw the Penrose triangle initially.

Data part 11) the Y-table

This was one piece of data I left intact. Since MM and JSW both use the Y-table the dual usage of the data was exactly what was required. 2 Y-tables combined into 1.

The code

Whilst the biggest chunk of memory was allocated to the data. The two games are not using the data in the same way. The size of data tables changes between MM and JSW and the game needs to have just one running game logic.

I briefly mentioned the data change from MM to JSW concerning the al-

location of graphic tiles. For a dual game it becomes cumbersome to adjust the code at run time to change between the various sets of data. It is quicker to run the code with both sets of game options running with no checks. In order to do this the room data is expanded with a piece of code that allocates undefined graphics with unique data values.

Explained in a different way. The code is set up to play both MM and JSW at the same time. When playing a MM cavern the code scans and acts on stairs. Since the set up code has inserted unique values into the look up code for stairs, because stairs are not defined in a MM room or its data. This ensures that the game code can not find any graphic that matches a stair. So no stair action takes place in MM. In a similar way when playing JSW. The collapsing floor/baddie 2 and the switch are allocated unique codes, which are not defined in the room data for JSW. The ensures that the code that acts of these types of graphic will not find the graphic in the JSW room plan. So whilst it checks to see if it should collapse a tile beneath Willie's feet. It will not find the tile needed to activate this action, as none will match the unique tile definition.

Expanding on the above. For the room data for either JSW or MM the game code sets in action an extraction sequence that fills in with unique data any data that is required, but not actually defined for either JSW or MM. The room game type (MM or JSW) defines what is allocated and what is generated. Easier to write a list of what is added to each game part.

For JSW we add these entries into the games room data. Knowing it will not be used.

1. Collapsing floor
2. nastie 2
3. switches
4. Willie's position and direction faced
5. air and timer values
6. portal positions and colour

for MM we add these entries into the room data. Knowing it will not be used

1. stairs (ramps)
2. exits left, left,right ,up and down

The addition of the data is created by a routine, that is used to extract room data. The routine is nearly always called forge in my code. And the version of FORGE in MJSW is a very large piece of code.

The code - forge

As stated above FORGE is a routine that is used to extract the room data. With Matthews room extraction code, that is all it does. For MJSW and most of the versions I write FORGE does a great deal more.

Things not needed in a single version game. (eg MM or JSW on their own)

1. As previously stated the data stream for room plan extraction varies in data size
2. JSW and MM rooms use differing types of graphics
3. JSW and MM use differing Item storage areas. Forge is used to locate the first item in any room played. The game loop does not search for items, the first item is thus already defined.
4. the flashing item graphic is pre defined in forge.
5. the start of a conveyor is pre defined in forge
6. game calculations for colours such as inks and paper, are pre calculated in forge.
7. room logic for things that are needed and unique. such as skylab, Maria, toilet animation. are pre defined in forge.
8. lower screen colours - pre defined in forge
9. Willie’s possible change into a flying pig - pre defined in forge.
10. Willie definition- as defined by play in JSW or MM
11. position of portals
12. graphic for portals

The above is just a small range of what forge needs to do. Its aim is not just the drawing of a new room and setting up the room data. (that was all Matthews version did) my version is also concerned with calculations that are done none stop in the game loop, which should be done once and only once per room. These are the needless calculations, that exist in the game and should be removed. A simple example. For each item flashed and drawn when playing, the item code loop calculates where the item data graphic is. The calculated answer never changes for every item drawn in a room. Yet that calculation is done on every loop for every item. - That task is now done once in forge and the answer written into the item loop. so the game loop calculation is removed.

The specifics of change are difficult to list, I tend to modify most routines. The routines that managed to stay unchanged are few.

-------- Run out of steam on describing changes

To ram home the part about additions to code being easy as long as the additions sit in the free space in the original code. Here I list my additions to the code. Which can not sit in the normal free space - that was long ago assigned to fit in another game.

Both games in 48k
portals in JSW
Block screen copy using LDI
interlaced colour on block screen copy
Willy scroll up on game ending

Final game screen with
Animated tea and slow text draw
Animated final barrier and title screen
reverse sprites
interactive Maria
interactive Nightmare room
Phase change on spinning Wabbit
colour change on light beam
illuminated beam hit on colour light beam
changing illumination on sprite collision for light beam
Speed change icon
speed change ability
Music on/off icon
Rooms visited statement
Constant high score update
display of high score on pause
Large 16 pixel high title scroll
4 pixel movement of title scroll
Removal of inter room graphics status flicker
Item count down.
Sound played for game completion
Animated Pac man worm for game completion
Policy bonus for game completion.
JSW portals
Changing game modes - room play,fixed,random
Items merge to screen - not erase as the original
Portals merge to screen - not erase as the original in MM
etc.

And all still in 48k