**Other Concepts**

**Animation (General)**

See Animation\_Manager.ASM

**Assembling Game Code**

1. open windows command shell prompt to c:\my\_code
2. execute go1.bat / go1hd.bat (it's actually in the c:\sbasm folder, but it should be in the system path)
   1. this batch file calls map.compression.ps1, a powershell script, to compress map data. The script puts the output files in c:\my\_code\includes\_libs\compressed\_data, where SBASM will look for them.
   2. then the batch file calls SBASM to assemble the game code
   3. the map compression utility is called to compress the map data (full size maps only, like the surface, no 32x32tile buildings maps are compressed
      1. Note: this step must be enabled, for each map. by default the maps don't get updated with each build to save build time.
   4. binary.collator.exe is called to pack and collate the binary output files for each NPC's speech text, which were generated by SBASM
      1. Note: this step must be enabled, for each building. by default the NPC speech text doesn't get updated with each build to save build time.
3. Press any key to continue when prompted to automatically run the 2nd batch file (go2p.bat / go2hd.bat)
   1. this batch file makes a backup of the game code, then copies the binary output created by SBASM into the AppleCommander Folder.
   2. Then it calls AppleCommander to add the binary output to the disk images.
   3. Then it copies the disk images to various application folders such as AppleWIN.

**Auxiliary & Bank Switched Memory**

Both are used extensively. For low level details in general on how these memory regions are used in the game, see my\_code/notes/my notes (memory\_mgt) and scroll down.

**Wormholes / Locations / Enterable Maps**

see subroutine docs and code for MAP.ENTER.CHECK (MOVEMENT\_MANAGER.ASM)

See code at MAP.EXIT.CHECK (MOVEMENT\_MANAGER.ASM)

**Graphics Screen Movement**

See Graphics Details.doc, take particular note of the "Key Variables" section.

See subroutine docs for:

MOVE.NORTH (Movement\_Manager.ASM)

SCROLL.COLUMNS & SCROLL.COLUMNS.DATA (Graphics\_Scrolling.ASM). Note the scrolling routines for ROWS have the same documentation.

**Loader Zones**

See Loader Zone Details.doc, and Zone\_Functions.ASM

Enabling/Disabling Compression: change table at start of powershell program and WZONE.COMPRESSION.FLAGS which is a lookup table in offload\_variables.ASM. There used to be another table in LOADER.P.ASM, but that has since been removed.

Compression flags will only be used if the powershell program call contains 1 as the 3rd command line argument. See $use\_compression\_flags in the powershell code for more information. If compression flags are not used then all zones are assumed to contain compressed data. And, USE.COMPRESSION\_FLAGS must be set before the call to ZONE\_TOOLS.BUILD.WZONE\_HEADERS (LOAD.NEW.MAP) or otherwise it will assume all zones contain compressed data.

Notes:

\*\*\*\*WARNING\*\*\*After changing compression status on a zone, regenerate all surface and undermaps (any map using the 128 tile x 128 tile structure), because the compression flags are used on all maps. Thus, if the maps aren't regenerated, there will be a mismatch between the data and the compression flags.

\*If compression enabled for a zone which has tile data that has a negative compression ratio, strange things can happen. I've seen it cause a lock-up when zone-transitioning into the region containing the zone. I've seen it result in most of the tiles in the zone being displayed as water.

\*If there are problems, double check the zone number of the zone that compression is enabled on. The zone flags are numbed in decimal and the zone number chart has hex and decimal numbers and I've mixed it up before.

**NPC TALK**

For datagram of NTALK.SEARCH.RETURN.SUB\_COMMAND.FLAGS. See one of the .tlk data files. These are the flags that indicated whether a given NPC response requires whisper, yell, or a particular event flag status to see the response.

To add subcommands see .SEARCH.SUBCOMMANDS (NTALK.SEARCH.LOOP)

NPC Speech Text is compressed using LZ4 compression. It is packed using LZ4.EXE during the PC-side build process and unpacked using an unpacker written by Peter Ferrie. The general process flow is this:

\*SBASM generates one binary output file per NPC

\*binary.collator.exe calls the compression tool to pack each binary file into a packed data bubble or chunk. Then it collates the binary files together into a single binary file for each town inserting the NPC ID and the length of the bubble (as unpacked values) at the front of each bubble.

\*SWAP.ROUTINES.Npc.Talk.ASM loads the collated packed binary file from disk for the town the player is in. It calls Peter's unpacker to unpack the speech bubble for the NPC the player is talking to. It figures out where to find the packed data for that particular NPC by using the NPC ID and length bytes at the start of each packed bubble

binary.collator.exe is a QB64 program I wrote, the source is locate in c:\my\_code\compression\npc.speech.text

**NPC Schedules**

See Chart 1.3, Chart 1.3a and the file and subroutine documentation in NPC\_BUILDING.MANAGER.ASM

Ladder Anchor Setup: See Building Map Spreadsheet, Design Notes worksheet.

**Water Animation**

Water animation has some unique characteristics. see subroutine docs for MOVE.COMMON.ROUTINE (Movement\_Manager.ASM)

**Special Effects / Features**

1. Quicksand:

see subroutine docs and code for .CHECK.QUICKSAND (MOVE.COMMON.ROUTINE, Movement\_Manager.ASM)

see code located at .SLOW\_PROGRESS.CHECK

(PLAYER.COLLISSION.CHECK, (Movement\_Manager.ASM))

see chart 1.1

see subroutine docs for TEST.TILE\_TYPE (Animation\_Manager.ASM)

see code located at .TEST.PLAYER\_LOCATION (TEST.TILE\_TYPE, Animation\_Manager.ASM)

see subroutine docs and code for ANIMATION.SCROLL.PLAYER (Animation\_Manager.ASM)

1. Swimming

see subroutine docs for MOVE.COMMON.ROUTINE (Movement\_Manager.ASM)

1. Crocodile:

see chart 1.1

see subroutine docs and code in ANIMATION.TILE.FRAME\_CYCLE, (Animation\_Manager.ASM)

See code in MOB.DRAWTILE.ENTRANCE2, (map\_objects\_management.ASM).

*See code .SPECIAL\_FLAG.COASTAL (COLLISION\_CHECK\_ONSCREEN,* map\_objects\_management.ASM)

1. Multi-Tile Mobs

see chart 1.1

see subroutine docs and code in DETERMINE.MOB.TILE\_TYPE, (Animation\_Manager.ASM)

*also code in map objects manager*

;For MTT S\_ENTITY, the S\_ENTITY type code stored in SCREEN.MO\_SPRITE\_TYPE.DATA

;contains a bit flag (bits 3-6) which identifies the position of the tile within the

;MTT S\_ENTIT (i.e. upper left, upper right etc).

;

;See ";===BIT FLAG MATRIX===" in MAP\_OBJECT.ADD.MTT\_FLAG\_BITS in the subroutine documentation

;for more details on these bit flags.

1. Multi-Tile Frigates

see subroutine docs and code in ANIMATION.TILE.FRAME\_CYCLE, (Animation\_Manager.ASM)

see subroutine docs and code in .TRANSPORT.DRAWTILE.MT (map\_objects\_management.ASM)

1. Double Mover MOBs

see subroutine docs in MOB.MOVE.MAKE (map\_objects\_management.ASM). The docs comment on where some of the code is located.

1. Tall Grass, Beds, Cots

see code in MOB.DRAWTILE.ENTRANCE2, MOB.DRAWTILE.MT (tall grass only)

(map\_objects\_management.ASM).

see code in .STANDARD\_ICON (DRAW.TILE.PLAYER), (graphics\_engine.ASM)

TEST.TILE\_TYPE (Animation\_Manager.ASM)

1. Horse Jumping

see code in NORTH,SOUTH,EAST,WEST,BOARD, JUMP (game\_loop.ASM).

see code in .HORSE\_RULES (PLAYER.COLLISSION.CHECK, (map\_objects\_management.ASM).

1. Sunset/Sunrise, Player Light Source (PLS), External Light Source (ELS)

see code in DARKNESS.TOD, DARKNESS.PLS, DARKNESS.ELS (darkness\_manager.ASM).

see code in CHECK.SUN.STATUS (event\_manager.ASM).

Add more sunrise/set exempt map or map types: see .CHECK.FOR.LOCATION.EXEMPTION (CHECK.SUN.STATUS, event\_manager.ASM)

Add more ELS tile types: search for "#DARK\_FLAGS.ELS.EQ1" in (DARKNESS.ELS) darkness.manager.ASM. There are a lot of maps because ELS is done via a fairly unrolled loop. \*\*\*4/8/17; last time I added a light source, I added some extra constants to the code which have place holder values in offloaded\_variables.ASM. If they haven't been used, just modify the values in offloaded\_variables.ASM.

1. Time / Display Clock

see code in TIME.DISPLAY, TIME.UPDATE.MOVE, TIME.UPDATE.DISPLAY (event\_manager.ASM).

**Battlefield Maps, Adding**

\*Create a new map in the "Combat Battlefield Maps.xls" spreadsheet.

\*Add a tile data hex table for the new map to the LOCAL VARIABLES section toward the end of the SWAP.ROUTINES.Combat.setup\_exit.ASM file.

\*Add branching logic to connect the pointer to the new map to .INIT.SCREEN (COMBAT.SETUP.MISC)

**Combat Maps, Adding**

See "Battlefield Maps, Adding"

**Cursor, update shape table**

See CURSOR.DRAW in map\_tools.ASM (graphics routine section). The cursor shape table is stored directly in this subroutine as .CURSOR.SHAPE\_TABLE

**Collision Rules**

-Adding Obstacles

.WALKING.TILE.TESTS, PLAYER.COLLISSION.CHECK (Player)

.HORSE.WALKING.TILE.TESTS, PLAYER.COLLISSION.CHECK (Player)

.SPRITE.WALKING.TILE.TESTS, COLLISION\_CHECK\_ONSCREEN (MOBS/NPCs, on screen)

.LOOP.DIRECTION\_GROUP, COLLISION\_CHECK\_ONSCREEN\_MTT (MOBS/NPCs, MTT)

.LOOP.MOB\_LAND, COLLISION\_CHECK\_OFFSCREEN (MOBS/NPCs, off screen)

IDENTIFY.OPEN.PATHS in NPC\_building.manager.ASM doesn't need collision rules because instead it has rules that identify Nox A\* grid tiles which are always passable.

\*Most rules are applied as CMPs against a tile type value loaded into the ACC once. When adding new rules be careful not to clobber the ACC value when it is expected to contain the tile\_type.

**Compression, Enabling/Disabling**

See Load Zones in first part of the file

**Doors**

Doors are created by adding a map object.

The unlocked/locked/magic\_locked status of the door is controlled by the value of byte $03 of the map object record. See map objects spreadsheet byte 3 for door status codes.

The terrain tile\_type should be the tile that will be visible when the door is open. The map object tile for the door should be the door tile that matches the status code. For example, if the status code is locked, then use the locked door tile\_ID. When an locked door is unlocked the tile\_ID is decremented by $1.

Doors are treated as obstacles via status codes; the tile\_ID of the door does not need to be in an obstacle tile\_ID range.

Window doors are obstacles unless the player is adjacent. Functionality is controlled by a tile\_ID range which is checked in .CHECK.WINDOW.DOOR (darkness\_manager.ASM). Currently this check is done for both BUILDING AND UNDERMAP together so windows doors must be in the same range for both.

There is a lot of code related to doors in the map objects manager. Also some in the darkness manager.

I've confirmed the following sections have code that relates to doors:

OPEN.COMMAND

.CHECK.DOOR (GENERAL.ENTRANCE) ;map objects, prevents drawing of open doors

**Dungeon NPCs, Adding**

See NPCs, Dungeon

**ELS, Adding**

See "Lightsources, Adding/Changing"

**Event Flags**

See .PROCESS.EVENT.FLAGS (NPC.SEARCH) for event flags processed DURING NPC conversations. Useful for doing things like having an NPC unlock/open a door as a result of saying something to them.

I might also add an event flag processing section to TALK.COMMAND in GAME\_LOOP.ASM just after the return from NPC.TALK (and after the memory swap in is done) for event flags that don't need to be processed during conversation. Or this could go in .EXIT in NPC.TALK, just after the main conversation loop. Either works, just depends on where memory is available. If it's done in NPC.TALK then the aux memory values need to be modified for any map object array changes as those values will get swapped back in after the return to main game loop.

See EVENT\_MANAGER.ASM. This routine is parsed on every iteration of the game loop

**Files, Adding to Build Process**

\*Create a new .ASM file via notepad using and existing file as a template, and adjust the name of the .BIN target file (the .TF directive is in the first few lines of the file)

\*Add the file to the floppy (go2p.bat) and harddisk (go1hd2.bat) batch files. There are two places to add the new file; toward the top where the files are being copied to C:\AC1.3.5 and further down where the files are being written to the disk image. The .BIN filename specified must match the .BIN filename specified in the .ASM file

\*Add the file to the FILES section (look for FILES heading) in offloaded\_variables.ASM. The ProDOS filename in this section must match the ProDOS filename specified in the batch files because that is the filename that is written to the disk image. Use the ProDOS filename in conjunction with the PRODOS.IO subroutine for file I/O.

\*Add an .INCLUDE for the new file in the "SEPERATE TARGET FILE" area of the INCLUDE section toward the end of game\_loop.ASM.

**Font, Editing**

\*\*Warning: the notes below were my plan. Everything is accurate except that the .set files on DOS Toolkit are not editable for some reason by the HRCG editor from assembly lines. As a result, I currently have the stock roman.set in the game.

\*\*update 5/17/17; reread these notes. I think this means that if we want a custom font we need to create all characters from scratch using the assembly lines editor. That sucks for making minor tweaks but if we want a font that doesn't have color bleed we have to make it from scratch anyway. though it may be worth checking every single font on the DOS toolkit disk just to see if one of those fonts is all white. I know I didn't try them all.

---start docs--

The Nox font file was based on C:\My\_Code\Fonts\misc fonts\roman.set, which was one of the fonts on the DOS Toolkit disk image. I edited this font set using the HRCG editor for which the code was posted for in chapter 32 of Assembly Lines Cookbook. I downloaded a disk image that had the program on it.

1) Launch disk image in my\_code/fonts/nox font editor

2) run nox\_editor (or load nox\_editor, "list", them look at REM instructions)

3) make changes and save.

4) In Ciderpress right click on the font file the changes were saved to, and select "extract". Keep Apple II formats.

5) rename the extracted file to nox\_fonts.set.bin and copy it to c:\my\_code where it will replace the existing font file that is used in the build.

**Increase Party Size / Add Characters (during testing, when party data is inserted via source code data files)**

Add new character to the following:

\*map objects file

\*character sheet file (place $AA in byte0 of the first empty record if any, used to auto-calc party size

\*readied equipment file

party size is calculated in .INIT.PARTY.SIZE (LOADER.P.ASM)

**Lightsources, Adding/Changing**

In the offloaded\_variables.ASM file, update the section in Darkness Manager for ELS tiles

search for ;DESIGNATE EXTERNAL LIGHT SOURCE (ELS)

If more preset labels are needed, to add more labels see documentation at the top of DARKNESS\_MANAGER.ASM

-Add conversation to dungeon NPCs

\*Follow the procedures in "NPC Talk, adding new .TLK file" and " NPC Talk, Adding New Record (existing .TLK file)" in this document. Nothing else needs to be done.

\*note that if the S\_ENTITY type is changed to dungeon NPC (see comments above), then that NPC type will need to be permitted in the entrance validation code for the NPC Talk module located in .CHECK.FOR\_NPC (TALK.COMMAND). Currently entry is only permitted if player tries to talk to a S\_ENTITY with the type code for building NPC.

\*\*\*WARNING: after doing the dungeon NPC test, I left the same talk file connected to the undermap level 1 map as is connected to the test town. Sharing the same talk file is possible for testing but NPCs with the same map object record index will end up with the same conversation text block.

**Location, New, Adding**

See Map, New, Adding

**Location, New Entrance, adding**

See Wormhole (between maps), Adding

**Map, New Entrance, adding**

See Wormhole (between maps), Adding

**Map, New, Adding**

\*add new map to map\_objects.xls in my\_code\maps\_shapes\map

("LOCATION LIST" heading, "Wormholes" worksheet)

\*Create new map spreadsheet: do as save as from template spreadsheet (see my\_code\maps\_shapes\map\templates)

-Create new data files

map : do save as from the template file (data.map.template.asm).

\*Make sure to specify the filename in the .TF directive.

sprite: do save as from the template file (data.spr.template.asm).

\*Make sure to specify the filename in the .TF directive.

\*Verify the map object arrays are set to blank (all $00s, except NPC schedule which needs $FF stop value as first byte).

talk: see "NPC Talk, adding new .TLK file"

\*Add data files to build process (see "Files, Adding to Build Process")

\*add new map to ";======FILES=================" section in offloaded\_variables.asm

\*Add new map to LOAD.NEW.LOCATION in (map\_tools.ASM)

\*Add new map to .IS.LOCATION.ENTERABLE (MAP.ENTER.CHECK)

\*Add a wormhole to the new map (See "Wormhole (between maps), Adding")

(;\*\*\*\*\*WARNING: !!!MAKE SURE STOP VALUE IS IN CORRECT BYTE OF MAP.WORMHOLES\_xx!!!!)

**Map Position, Starting**

Set GMAP.X/Y in .MAP\_POSTION.SETUP.

Note: GMAP.X/Y can be set to any location on the map that isn't beyond the zone transition column/row in edge zones)**.** (i.e. it is not required that GMAP.X/Y be set to a zone's center tile)

**Master Price List**

See Merchant (shop), price management

**Merchant (shop), Adding**

Shop merchants (which use the INV\_8 merchant interface sub\_module of the inventory file) are identified as being shop merchants by their tile\_ID (see COMMAND.TALK), and their particular merchant profile is identified using their NPC record number and the map code of the map the exist on (see .INIT.READ.MERCHANT\_PROFILE\_DATA (INV\_8.MERCHANT\_TRANSACTIONS))

-Adding a Shop Merchant

Add an additional profile record to "TEMPLATE merchant profile" in 3Combat\_Inventory.stats.xls and paste the hex data into the source code file data.game.merchant\_profile.ASM

**Merchant (shop), price management**

Prices are set via the 16-bit price field in the master item tables. For example, in the weapons table, see column R.

The price data is propagated to "TEMPLATE master price table" in 3Combat\_Inventory.stats.xls. Copy and paste the hex data source code into the file data.game.master\_price\_table.ASM

**NPCs, Dungeon, Adding**

-Add a record to the NPC array in the SPR data file for the dungeon.

\*The NPC should automatically appear. There is entry validation code in .LOAD.NPC.RECORD (map\_objects.manager.ASM), but the undermap map type is permitted.

\*I haven't tried it but I think if byte $6 (in-transit flag) is set to $00 (not in transit), then the movement options for byte $7 (at-anchor movement flag) will work.

\*I think that map\_objects.manager.ASM is setting the S\_ENTITY type to building NPC. That might be fine but I did setup a dungeon NPC type. I think S\_ENTITY is set by map\_objects.manager.ASM during the initial screen draw. So to set type to building NPC it probably requires a map type check in the routine which sets the S\_ENTITY type.

**NPC Talk, adding new .TLK file**

CURRENT.LOCATION.TLK\_DATA is a variables that stores the memory address which contains the filename of the .TLK file of the location that the player is currently in.

Set CURRENT.LOCATION.TLK\_DATA just after the sprite data file load (after the call to PRODOS.IO) for the associated location in LOAD.NEW.MAP. Search for "CURRENT.LOCATION.TLK\_DATA" for an example.

**NPC Talk, Adding New Record (existing .TLK file)**

\* Add the conversation text to the .TLK file for the building the NPC is in, making sure the record # in the .TLK file matches the record # in the .SPR file. In the .TLK file, make sure to specify a new .BIN file for that conversation bubble. This is done via the .TF (target file) directive at the top of the conversation bubble. Look at an existing record and copy/paste to get on the right track.

\*That's it. No need to add the new .BIN file to the build batch files because the individual .BINs are collated by a QB64 program and only the single collated .BIN file is added to the disk images.

**Obscuring Tiles, Adding/Changing**

See Lightsources, Adding/Changing

**Player Icon: setting**

It is done by changing the value of the LDA in the appropriate section of GAME.START.DRIVER (2nd MIDDLE section). Currently the label name of the associated STA is PLAYER.WALKING.TILE

**Prices**

See Merchant (shop), price management

**Portcullis & Portcullis Lever**

See OPEN.COMMAND (game\_loop.ASM). There might be more areas.

Setup as general map object pair, with the portcullis lever in the map object record after the portcullis

Visually, here is the layout:

(portcullis record, portcullis lever record)

\*portcullis record: set the flag byte to up (raised) or down (lowered). The tile\_ID is the same for either because when the up (raised) setting is the same flag value that triggers tile draws to be skipped (used for other features such as NPCs standing on ladders).

\*portcullis lever record: set TILE\_ID and flag byte of the lever set left/right depending on whether you want the starting position of the portcullis to be open/closed.

See general map object flags for portcullis flag byte settings.

Note:

\*The TILE\_ID of the portcullis and lever shape tables do not need to be the same in every tile set.

**Skills, Adding**

\*Add a skill field and skill progress field in the PC character sheet

\*Add code to increment the skill progress upon a certain event. For example, certain combat skills are incremented in .SKILL\_PROGRESS.UPDATE (COMBAT.HIT\_MISS.ROLL)

\*Add code to .CHECK.SKILL.THREASHOLDS (COMBAT.PROCESS.LEVELUP), which converts progress points into skill points when the progress bucket is full.

**Spells, Adding**

In .PARSE.SPELL\_CODE in SWAP.ROUTINES.cast\_spell.setup..ASM, add a branch for the new spell code added.

The branch will then call an entrance routine for the spell. The entrance routine will then call the code block for the spell, located in SWAP.ROUTINES.combat.spells.ASM. Walk through these steps in an existing spell and emulate the approach.

In the " ;=====SPELL FILE OFFSETS====" offsets section, add offsets for the new spell. Copy/paste the offsets for an existing spell and change the labels to reflect the name of a new spell.

The documentation at the top of SWAP.ROUTINES.combat.spells.ASM covers a lot of details such as:

\*where shape data is stored

\*how target hits are reported

**Sunrise/sunset transition**

**(day/night tile transition)**

Originally day/night tile transitions were handled by actually swapping out the contents of the shape table in AUX memory. I changed it later on to be managed via a terrain tile\_ID swap, which works the same in principle as a map object tile\_ID swap.

The terrain tile\_ID swaps are handled in .DAY\_NIGHT.TILE.SWAP (DRAW.TILE.TERRAIN\_ENTRANCE) in misc.main\_memory\_only.ASM

**---original method (shape table swap)**

See EVENT\_MANAGER.ASM documentation. Routines involved:

TILE\_SWAP.INIT.SUNRISE\_SUNSET

CHECK.SUN.STATUS

TILE\_SWAP.SUNRISE\_SUNSET.SCREEN\_DRAW

TILE\_SWAP.SUN

.LOCATION\_TYPE.BUILDING (LOAD.NEW.LOCATION)

GAME.LAUNCH

.LOAD.MISC.SHAPES (LOADER.P.ASM)

"MISC SHAPE TABLES" in LOADER.P.ASM

**Storms**

Current Status: there is nothing to prevent storms from moving off the edge of the world map. The result would be their GMAP X or Y flipping over/under to $00 or $FF.

If we stick with the current size world then add some code to MOB.MOVE.NORTH\_EAST to detect the map edge and then zap the mob or reset to other side. Otherwise it would take awhile for the storm to reappear.

If we go with a 256x256tile map then the flipping under/over behavior would result in the storm reappearing on the other side of the map.

Glitch: if the player chases a storm by moving north and east, in that exact sequence over and over, the player can keep up with the storm regardless of transport. Otherwise the storm can always outrun the player because it is moving at a diagonal.

Onscreen Movement

.CHOOSE.COLLISION.SUBROUTINE (MOB.MOVEMENT, map\_objects\_management.ASM)

MOB.MOVE.NORTH\_EAST (map\_objects\_management.ASM)

Offscreen Movemnt

.STORM.CHECK (SPRITE.ENTRANCE, map\_objects\_management.ASM)

MOB.MOVE.NORTH\_EAST (map\_objects\_management.ASM)

Animation override (if storm, draw animation over player icon)

.CHECK.STORM1 AND .CHECK.STORM2 (TEST.TILE\_TYPE, ANIMATION\_MANAGER.ASM)

.CHECK.STORM (DRAW.TILE.PLAYER, graphics\_engine.ASM)

.CHECK.FOR.MOB (animation\_manager.ASM); switches animation frame order based on bit test. The concept is that the storm objects are setup with different flag byte values so that the bit test yields different results, causing the lightning bolts between some mobs to flash at different times instead of being in sync.

**Unoccupied/Occupied Tile Swaps**

see code in MOB.DRAWTILE.ENTRANCE2 (map\_objects\_management.ASM).

see code in .STANDARD\_ICON, DRAW.TILE.PLAYER, (graphics\_engine.ASM)

see code in TEST.TILE\_TYPE (Animation\_Manager.ASM)

\*note: in TEST.TILE\_TYPE occupied tiles swaps should only be exempt if the occupied tile is not animated (i.e. beds)

see code in TEST.TILE\_TYPE (Animation\_Manager.ASM)

.TEST.PLAYER\_LOCATION (Animation\_Manager.ASM)

\*note: this section is used if the occupied version of the tile is animated

**Wormhole (between maps), New, Adding**

see documentation in WORMHOLE.DATA (offloaded\_variables.asm ) code section