Ships Components

1) MBD NAME FOR SHIP
SIREN

2) SHIP DATA FILE
HAS INFO ABOUT MESH FILE

3) SHIP MESH FILE

4)

#1 checkout Med.cfg

#2 DOS cd data\mesh

- copy existing mesh to new name of mesh
  ex: copy MULTI.IFF SIREN.IFF
- cd\dsrc\ships
  ex: copy wasp.xmf siren.xmf 8 letters
- edit siren.xmf
- change comments + "Base" to "siren"
- Save + exit

edit mdS.bat (check out)

- make icon. c:\wcs\util\missedit\cfg\siren.ico
- util\missedit edit med.cfg change shipstuff

mesh name or shipfile name = 8 characters Medname = NO SPACES

Wing Commander Armageddon
Series Overview

Series: Simulator Training Missions

Series Objectives / Description

There will be 5-10 Generic Missions. Mission Names S1-Sx
There will be 5-10 specific ship missions. Mission Names SS1-SSx
There will be 5-10 Multi-Player Scenarios for the players to choose from. Mission Names SM1-SMx.
There will be 5-10 Gauntlet-type missions. Mission Names SG1-SGx.

The objective of the simulator missions is to become intimately familiar with most flight, combat, and game elements seen in the regular missions throughout the "real" game, but with no enemy threat or plot-driven pressure.

The naming convention is as follows:

- The **Generic missions** will teach the player how to operate the ship in stages of complexity. The first will teach enough to get through the first three or four missions without needing to learn anything new, the second will test him on his ability to do basics. The pattern continues from there, presenting progressively complex information and testing the player's skills to that point. And the final stages of training will be optional elements of the cockpit (e.g. power control).
- **Specific Ship missions** will train the player on the advantages and disadvantages of each particular ship as it becomes available.
- **Multi-Player scenarios** will put the player against varying degrees of enemy ships.
- **Gauntlet missions** will put the player against varying degrees of enemy ships. Only Kilrathi Gauntlet modes will be available at first, but around Series E the alien ships will become available.

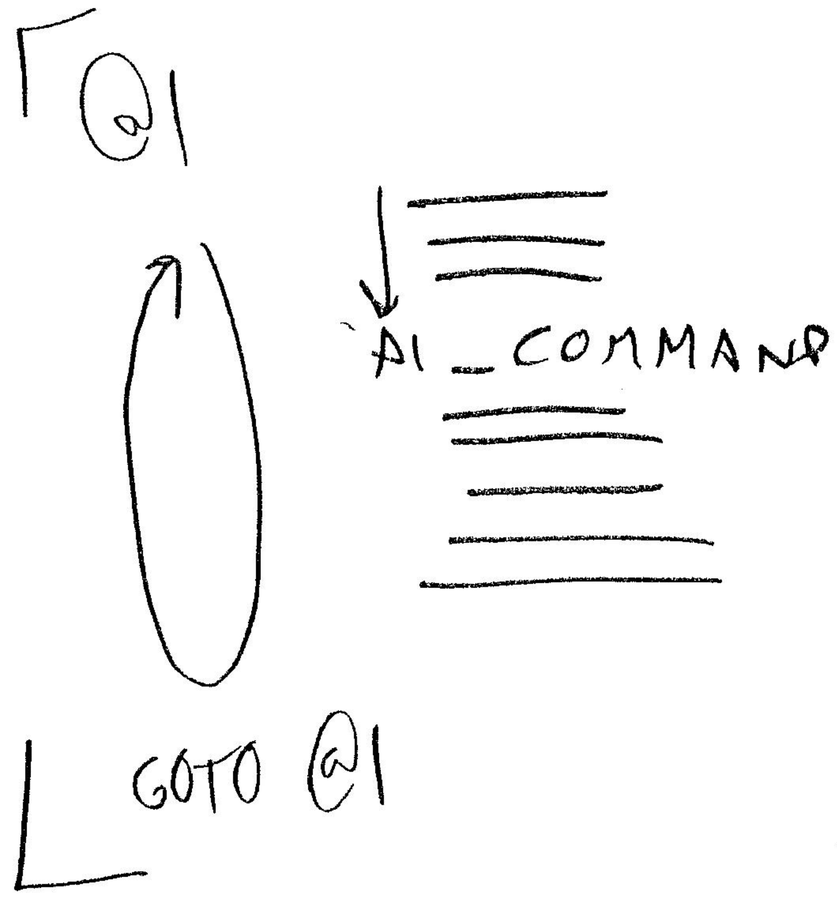
All of the training commands will be delivered by a "trainer", who will provide voice-overs. They will deliver more than adequate amounts of information.

The actual narration will vary depending on the peripherals the player has installed (joystick, keyboard, rudder pedals, etc.). See the mission-specific descriptions for details.

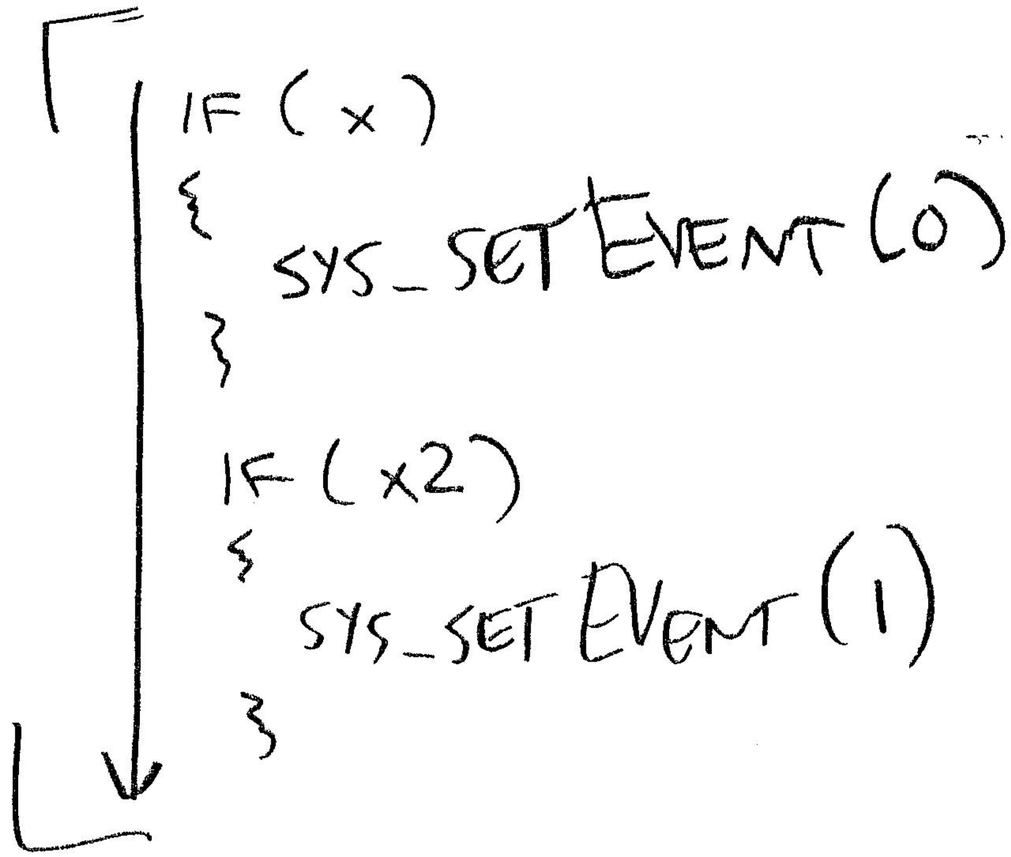
If the player ever wants to repeat one of these missions, he can.

We will probably give the player a small trinket, such as a sub-campaign ribbon (Wing I style), for completing all of the simulator missions.

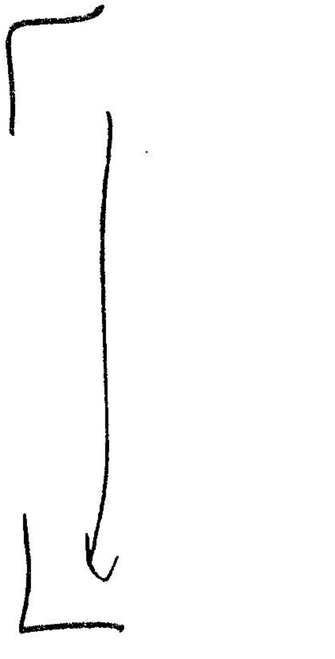
PROGRAM



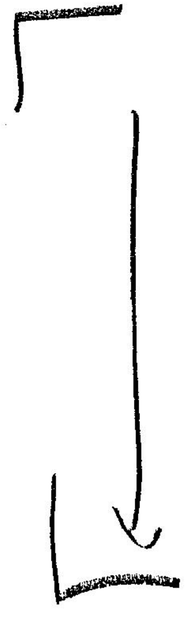
EVENT HANDLER

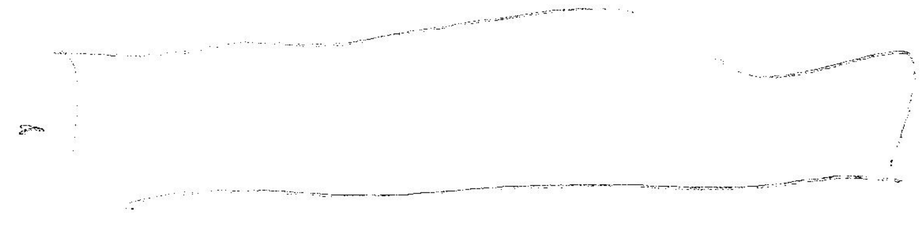


EVENT0



EVENT1





PROGRAM

```
@1
    AI_WAITSECONDS(10)
    GOTO @1
```

EVENT 0

```
    AI_ATTACK(PLAYER)
```

EVENT 1

```
    PLAYSFX
    (COMM1)
```

EVENT HANDLER

```
IF(AI_GETDISTANCE(PLAYER) < 100)
{
    SET EVENT(0)
}
IF(AI_GETDISTANCE(PLAYER) < 500)
{
    SET EVENT(1)
}
```

Cain, Billy

From: Sommers, Andy
Sent: Tuesday, May 27, 1997 10:57 AM
To: Hughes, Jason; Barnes, Cinco
Cc: ML WCP DESIGN
Subject: RE: Thoughts on Object Program Economy...

SYS_Message has been added for you (through the message plaque system) to print the value of any flag or inform you that you are executing a block of code WITHOUT fataling.

To test the randomness of "random", you should be able to use the following added lines in your program below :

From: Barnes, Cinco
Sent: Saturday, May 24, 1997 10:32 AM
To: Sommers, Andy; Hughes, Jason
Cc: ML WCP DESIGN
Subject: Thoughts on Object Program Economy...

We've been informed that limiting the number of unique object programs is a good thing for frame rate.
I'm trying to take this to its logical extreme.

The idea is to have one program that does a random roll for target importance that ***each object using the program*** will get a *somewhat* different value for. (Rather, that each object will get its own *random* number.)

However, I can't seem to generate the results that I need.

Please help me by taking a look at the following med code:

```
void M_EnemyFighter ()
{


// Enemy Fighter startup is standard.
// However, the ships will all be bound to 0 so that the player can
// have them "on his ass" the whole time!


  AI_WaitUntilActive();

  SF_SetObjectFlag(OF_vulnerable, 0 );
  SF_SetObjectFlag(OF_alignment, ALIGN_ALIEN );

  SF_BindToActionSphere(0);

  AI_ClearAttackList();


// To generate the attack list, each object that runs this program
// should get its own value for target importance.
// However, "Random" can generate zero, so we add ( 1 ) to avoid
crashing.
```

```
int temp1 = SYS_Random (100) + 1;
int temp2 = SYS_Random (100) + 1;
SYS_Message (temp1);
SYS_Message (temp2);
AI_AddToAttackList( Alpha2 ,   temp1 );
AI_AddToAttackList( Player ,   temp2 );


while ( 1 )
{

     AI_Attack();

}

}
```

The major problem for me is that the random values for target importance appear to roll *once* total.
Let's say three active ships have the above program. After they roll, they ALL go attack the same object.

If you're frowning, shrugging, with palms to the sky saying "What the fuck did you think this would do, Cinco?"
I would then like to ask that there be some way of getting individual objects to roll individual random values when hitting a program like this. I don't know if this would defeat the purpose, or if it would even be possible.... but I'd like to know.

Thanks!

- Cinco

DIFF TYPES of C
- EACH COMPONENT
- TRAPS / PITFALLS
- DEBUGGING TECHNIQUES

- ☐ TYPES
- ☐ OPERATORS
- ☐ EXPRESSIONS
- ☐ STATEMENTS
- ☐ CONTROL
- ☐ FUNCTIONS
- ☐ STYLE
- ☐ TRAPS
- ☐ DEBUGGING

TYPES – DATA REPRESENTATION

INT SAME AS LONG

LONG Value from −2147483648 to 2,147,483,647
DON'T USE LONGS

VOID

MOD/WCP USES ONLY INT, LONG + VOID

```
void FUNCTION001()
  {
    INT TEMP;
  }
```

void is a type with no value NOT ZERO – empty type.

[INT BOOLEAN = {0, 1}] MED DOES NOT SUPPORT!!

OPERATORS — characters to allow ~~type~~ manipulations on types

OPERAND — SOMETHING THAT IS OPERATED ON.
(TYPE OR VALUE OR RESULT OF SERIES OF OPERATIONS)

3 Types

UNARY works on one operand

- NOT "!" NOT

int temp

temp = !temp
Reverses State
If 0 => NON-Zero (1)
If 1 => zero

- "—" negation

switches sign of value
if negative => pos
If pos => negative

```
int temp=7;
temp = -temp;
temp == 7
```

- "=" assignment

```
Int temp;

temp = (-temp);
temp = 7;
```

- "+=" "-=" "*=" "/="

Increment

a=a+1 — "++" int a; int b;
b=b-1 — "--"

a += b; a=a+b;
SAME THING

Decrement

BINARY works on 2 operands

can't work by themselves

- "+" addition

```
int temp;
temp = 8 + 7;
```

- "-"
- "*"
- "/"

go from left to right

```
int a;
int b;
int c;
c = a + b;
a - b;
a * b;
a / b;
```
— divide by zero error

RELATIONAL

Comparison

BINARY OPERATOR

"&"

&	TRUE	FALSE
TRUE		
FALSE		

RELATIONAL

[&&] "logical AND"

BOOLEAN OPERATION

0 = FALSE

>-1 <1 = TRUE

&&	TRUE	FALSE
TRUE	TRUE	FALSE
FALSE	FALSE	FALSE

money

RETURNS 0

```
int a;
a = 0;
int b;
b = 0;

c = a && b; → c == 0
```

⋮

```
if (a && b)
```

RELATIONAL

"Logical OR"

||	true	False
true	true	true
False	true	False

$$

Money or ID
gets into club

BINARY RETURN 0 or 1

ID

```
if (FIRST KILLED || Vaquero killed)
```

"==" "equality"

(1)
returns true if both operands are equal
returns (0) if they are not equal to each other

```
int a;
a = (7 == 7)
```

a==1

```
if (shuttledocked == 1)
{
}
```

[!=] "INEQUALITY" or "Not equals"

returns 0 if both operands are equal
1 if both operands are not equal

```
a = (7 != 8);
```

RELATIONS

[>] Greater than
[>=] Greater than or equal to
[<] less than
[<=] less than or equal to

return a true(1) or false(0)

```
a = 7>8;      0
a = 7>=8      0
a = 7<8       1
a = 7<=8;     1
```

EXPRESSIONS

A collection of operators and (or) operands that can be evaluated

(7 > 8) (a) (!a) (5) (a==3) (a+b)

COMPLEX EXPRESSION: AN EXPRESSION COMPOSED OF MORE THAN ONE EXPRESSION.

```
int a;      a=7;
int b;      b=(a>=5) && (a<=10);
            b=1
```

STATEMENT A LINE OF CODE (THAT CAN BE EXECUTED) WITH ONE OR MORE EXPRESSIONS

YOU MUST HAVE A SEMI COLON AT THE END OF ANY STATEMENT LINE.

int a;

b = (a < 22) || (a > 27);

true

0 > 21 28 < 4

b true
b false

≥ 21 ≤ 28

~~219~~

b = 1 if b ~~a~~ ≥ 21
or
b is true if b ~~a~~ ≤ 28

~~b~~ a ≤ 21
~~b~~ a ≥ 28

b is false if a = 22, 23, 24, 25, 26, 27,

---

int a = 7;
int b = -3;

a = (b*7) / (a - a);

a = 21 / 0;

DIVIDE BY ~~0~~ error

~~a~~ [illegible] ←

no, What is a? = won't compile

OK why because you're dividing by zero

OK WCP = Divide by zero error in function001

CONTROL

How is your program going to execute?
PROGRAM FLOW

LABEL/GOTO

IF/ELSE

WHILE

LABEL/GOTO

LABEL IS A ~~COLLECTION~~ STRING OF CHARACTERS ≤ 4
FOLLOWED BY A COLON [ALPHA NUMERIC with ALPHA FIRST]

LOOP: ✓ 1LOO: X 1CATT: X

GOTO [LABELNAME];

Block of CODE

{ ← START OF CODE Block

// ——;
// ——;
// ——;

} ← END of CODE Block

IF/else

IF (expression)

MUST HAVE TWO

{
↓
}

// expression evaluated TO True

```
if (expression)
{
    // True
}
else
{
    // expression evaluated to FALSE
}
```

```
if (expr1)
  {
    // 1 but dont care about 2
      if (expr2)
        {
          // Both 1 & true
        }
      else
        {
          // 1 not 2
        }
  }
else
  {
    // NOT 1 but dont care about 2
      if (expr2)
        {
          // 2 but not 1
        }
      else
        {
          // neither are true
        }
  }
```

while Loop construct

is true

```
while (expr1)
{
  // code is executed
  // as long as expr1 is true
}
```

BREAK "Get me out of this while loop"

STYLE

PASS AT 4

1) USE COMMENTS

2) AT MOST ONE STATEMENT PER LINE
   2.1 [illegible]

3) NO COMMENTS ON SAME LINE AS CODE

4) PLACE COMMENTS (IF ANY) BEFORE STATEMENT OR CODE BLOCK

5) INDENT COMMENT TO MATCH INDENTATION OF STATEMENT OR CODE BLOCK

6) BRACES SHOULD BE ALIGNED w/ IF / ELSE / WHILE / FUNCTION

7) USE THE FOLLOWING NAMING CONVENTIONS

G_* ⇒ Function in MAIN } GLOBAL
g_* ⇒ Variable (flag) in MAIN }
M_* ⇒ Usage function in MODULE } MODULE GLOBAL
m_* ⇒ Use variable in module
all others ⇒ local variables

8)

TRAPS + PITFALLS

1) "=" vs "==" [illegible] operator

1a) "&" vs "&&" "

1b) "|" vs "||" "

2) misleading comments — make sure your comment matches code

3) number(int) overflow — 2.b1 to 2.b.1

4) uninitialized variables

5) all variables should have a unique name [illegible]

6) [illegible]

7) [illegible] all cases

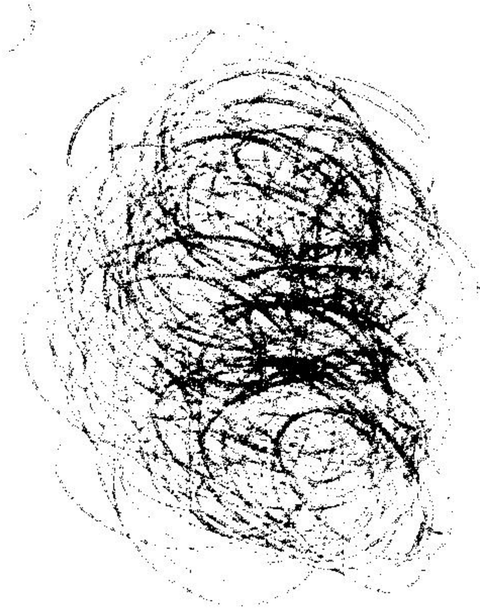
8) [illegible] conditions

9) [illegible]

DEBUGGING

TECHNIQUES OF 1965

SPECTACULAR	SOMEWHAT
[illegible]	[illegible]
[illegible]	[illegible]
[illegible]	[illegible]



1) [illegible] Perfect code to start with.

2) Shift to Finding Bugs
   A) Identify Bug Symptoms
   B) Isolate the Location
   C) Step through the code, line by line
      1. Verify that it is correct
         a. If line is wrong, a single statement, go back down the line

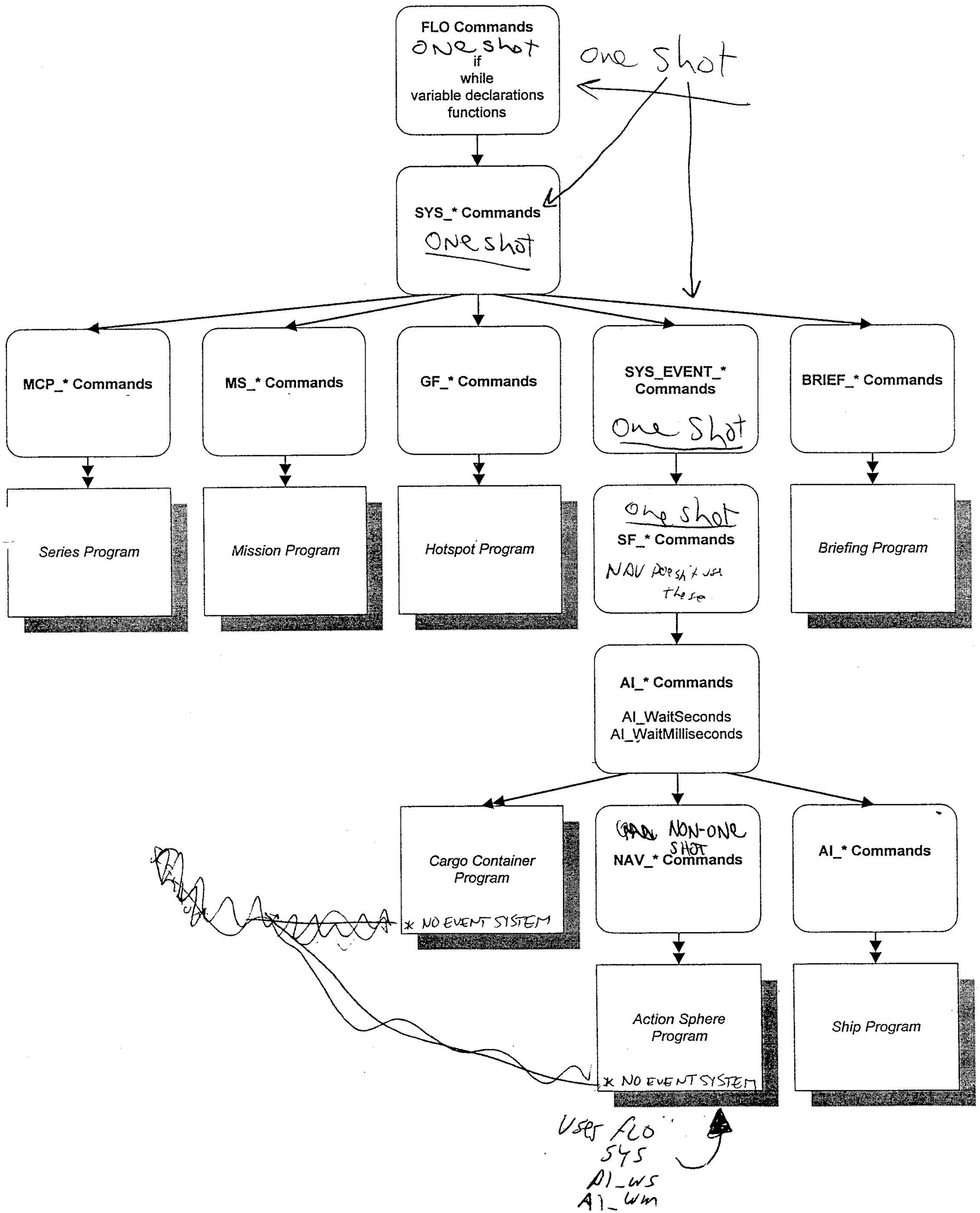
2. Fix problem with minimal possible [illegible] Side effects. Do [illegible] ([illegible] points, [illegible])

D. Prepare for a Simple Plan —

E. Get A Second Opinion

Wing Commander : Prophecy

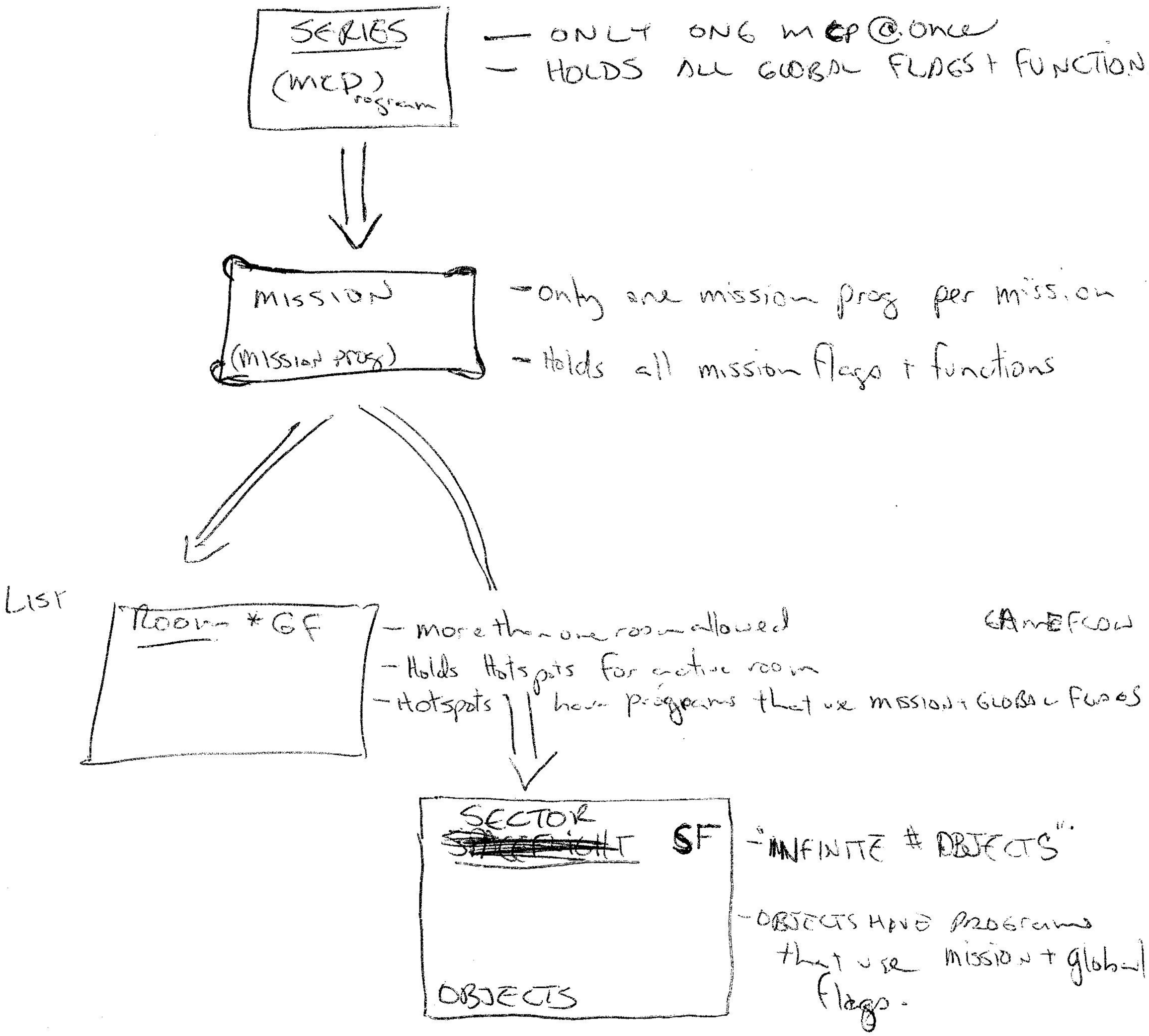
Mission System Program Hierarchy



MED CLASS

- SERIES
- MISSION
- ROOM
- SECTOR

WING PROPHECY TREE



M1

— MED CLASS —

SERIES

- ☐ SERIES PROGRAM
- ☐ LIST OF GLOBAL FLAGS
  NO LIMIT ON NAME SIZE
- ☐ LIST OF GLOBAL FUNCTIONS
  LIMITED TO 31 CHARACTERS
- ☐ LIST OF MISSIONS
  (BILLY ONLY)

SERIES PROGRAM:

- MCP_* COMMANDS:

```
MCP_RunMission(MISSION_61);
```

- SYS_* COMMANDS

```
int Sys_Clock();
```

Purpose: Returns current time from start of game in seconds

USAGE:

```
[var] = SYS_CLOCK();
int clocktime = SYS_CLOCK();
if (SYS_CLOCK() = whatever)
{
}
```

```
void SYS_Fatal (int error_code);
```

purpose: crashes game with specified error code (for debugging)

usage:

```
SYS_fatal(100);
```

void SYS_PlaySfX(int wavefile);

Purpose: Play sound effect denoted by wavefile
USAGE:

SYS_PlaySfX(37);

void SYS_PlayMovie same as above with movie

int SYS_Random (int max);

Purpose: Returns Random number from 0 to (Max-1)
USAGE:

int n=SYS_Random(37);

returns n => 0...36

Every Program has a timer associated with it

void SYS_TimerSet (int sec);

Purpose; Set "Program" TIMER (timer counts down)

SYS_TimerSet (3); // Timer is set to 3 seconds

int TimerGet ();

Purpose: Reads "Program" timer (timer counts down

SYS-TimerSet(3);

int timerVal = SYS_TimerGet();

if (SYS-TimerGet () < 10)

```
void    M_FighterMain ()
  {
    SYS_SetTimer(60);
    while (SYS_TimerGet())
      {
        pi wait milliseconds(250);
      }
    SF_DeactivateSelf();
```

int SYS.TimerZero ();

Purpose ; Returns True if Timer expired or = zero

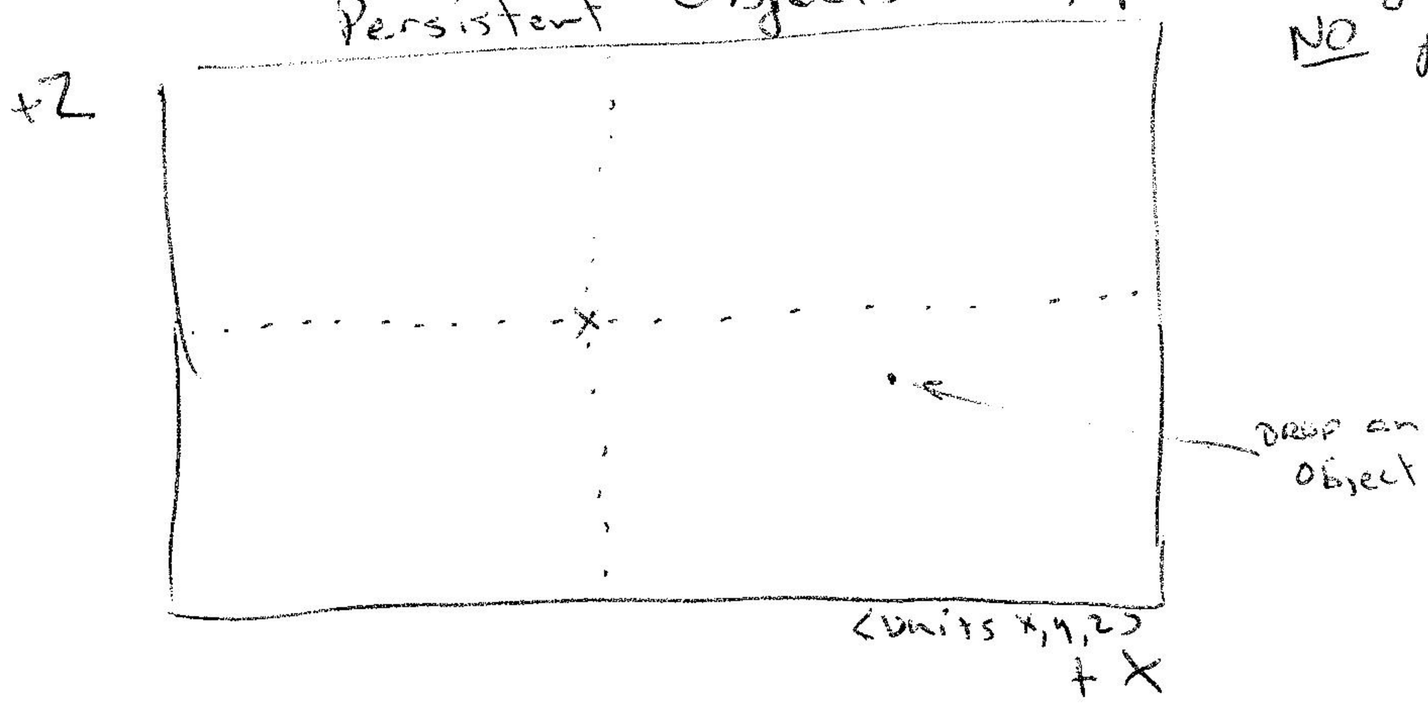
USAGE :

Med Class Day 2

- ☐ SOLIDS
- ☐ MISSION
- ☐ ROOM
- ☐ SECTOR

SECTOR

Persistent Objects - only place objects with NO programming.



- ☐ SHIP
- ☐ AST Belt Start
- ☐ AST BELT CONTROL
- ☐ CARGO
- ☐ DEAD OBJECT
- ☐ DEBRIS FIELD
- ☐ EJECTED PILOT
- ☐ HEAVENLY BODY
- ☐ JUMP POINT
- ☐ LIGHT SOURCE
- ☐ MINEFIELD
- ☐ NAVPOINT
- ☐ TURRET

ALL OBJECTS START AT +Z

A. LIGHTSOURCE (3 per ACTIONSPHERE)

ObjType DO_NOT_MODIFY {OT_lightsource}
Intensity {r, g, b} (0-1)
Orientation {y, p, r} (0-360)

Radius {klicks}

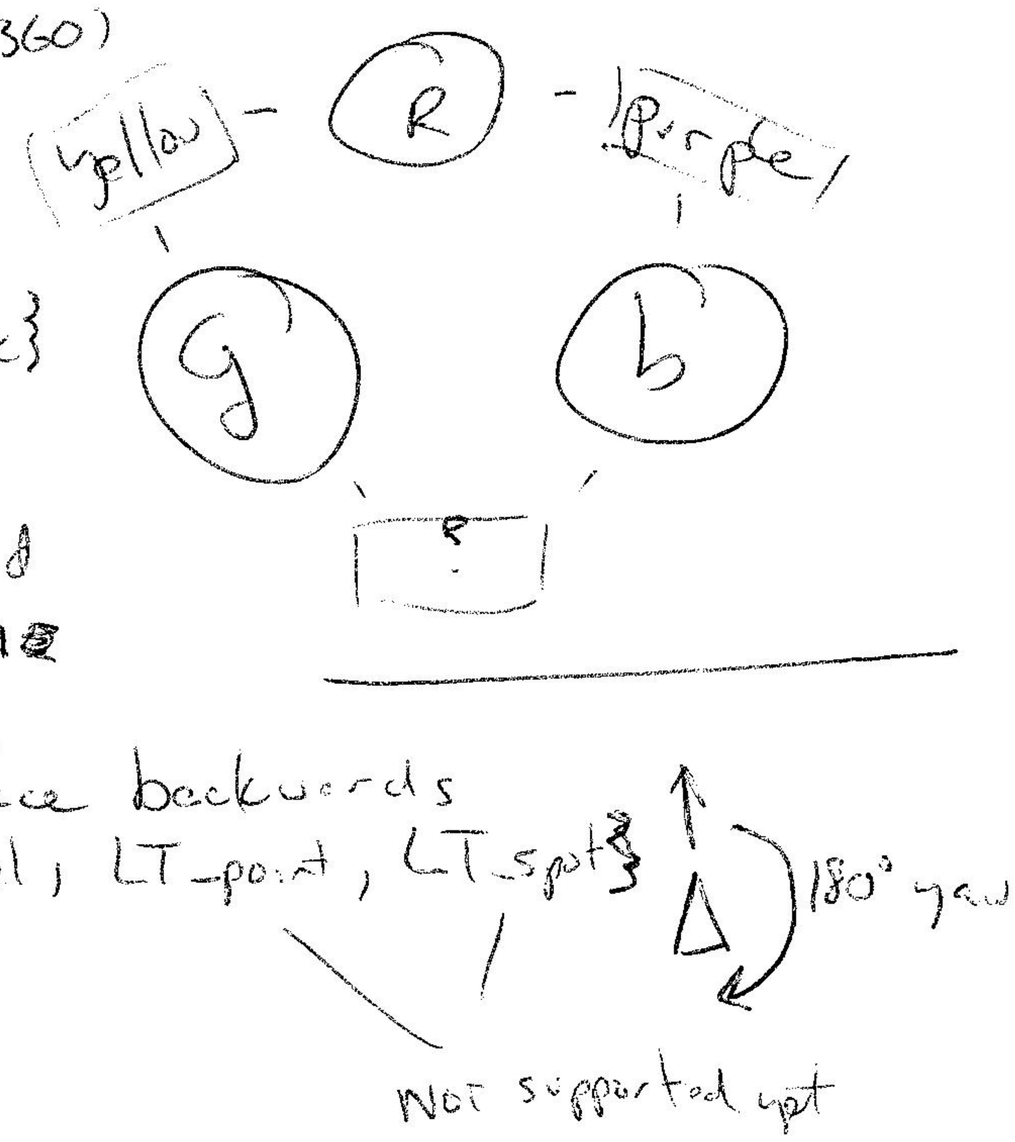
Rotation Rate {y/second, p/sec, R/sec}

degrees/second

Start NavPoint {n} 0...9

yaw 180 = Face backwards

Type Light {LT_Ambient, LT_parallel, LT_point, LT_spot}



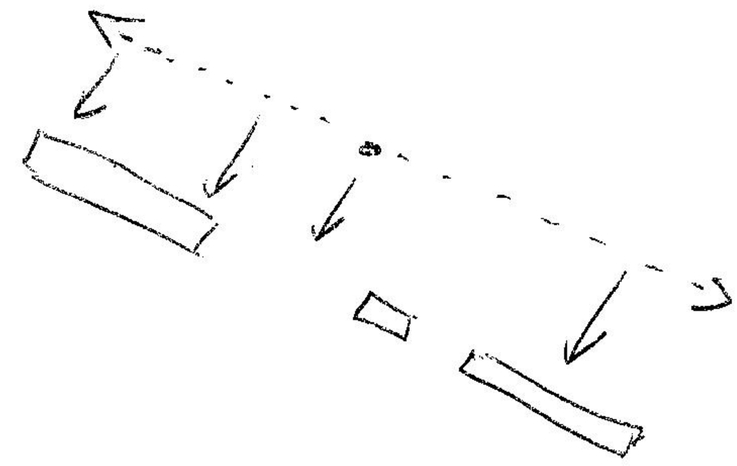
NOT supported yet

LT_ambient – intensity
Start New Point

- LIGHTS ALL SIDES OF OBJECT AND IS OVERALL LIGHT VALUE IN ACTION SPHERE
- 0.2 ... 0.4 are good values (Proposed)

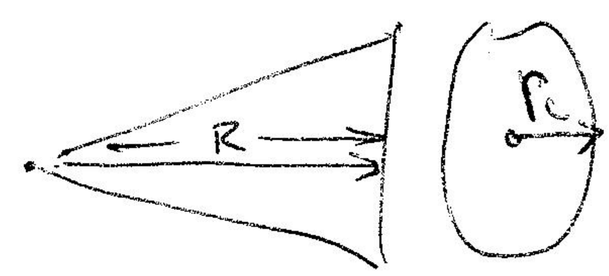
Average RGB R+G+B/3 = 0.2 to 0.4

LT_parallel – Intensity
(directional)
Start New Point
Orientation
Rotation Rate



LT_point – Intensity
Start New Point
~~Orientation~~
Radius

LT_Spot – Intensity
Start New
Orientation
Radius
Rotation Rate
Start New Point



Med Class

Asteroid Belt Start + Asteroid Belt Control Points

PROPERTIES

- _ObjType_DO_NOT_MODIFY
- Briefing Data
- Field Info
  - number (Field) ⟹ which field does this belong to (0...3)
  - number Items ⟹ number of asteroids in field (1...2000)
  - random Seed ⟹ seed to initialize random Number generator
  - yps ⟹ max yaw/sec for an asteroid
  - pps ⟹ " pitch/sec " " "
  - rps " roll/sec " " "
  - radius ⟹ radius of crazy straw "cylinder" segmentedness

NOTE - ONE AST BELT START PER AST FIELD
- AT LEAST ONE CONTROL POINT NEEDED
- UP TO NINE ADDITIONAL CONTROL POINTS

Ast Belt Control Points

Properties
- ObjType - DO_NOT_MODIFY
- CONTROL POINT
  - Field number ⟹ which field does this control point belong to?
  - cp-index ⟹ " control point am I in the field

DEBRIS FIELD
MINEFIELD

NAV POINTS (SAMEAS ACTIONSPHERE)

"CONTAINER FOR OBJECTS"
"O is persistent action-sphere"

Properties
- Ob_Type_Do_NOT_MODIFY
- Briefing Data
- DescNav {String}
- FLO_Main "FILL THIS IN"
- Nav Point
  - nav index {01 . . 9}
  - hiddenflag {true or false}
  - radius
  - drop off range

- Nav point number in Nav Point List
- whether or not NavPoint shows up in briefing
- in klicks NavPoint Radius
- Distance from Center along entry vector..

NUMBER ~~Place~~ all NON-HIDDEN NAV POINTS FIRST

---

void NAV_ActivateSelf ();
purpose: To have an actionsphere 'turn on' and start running the programs for ~~ships~~ objects attached to this Nav.

void NAV_DeActivateSelf ();– turn off and Pause programs

~~void~~ int NAV_WithinSphere ~~(_ObjId o)~~ (_ObjId o);
purpose: returns true if ~~ObjId~~ o is within the action sphere defined by radius

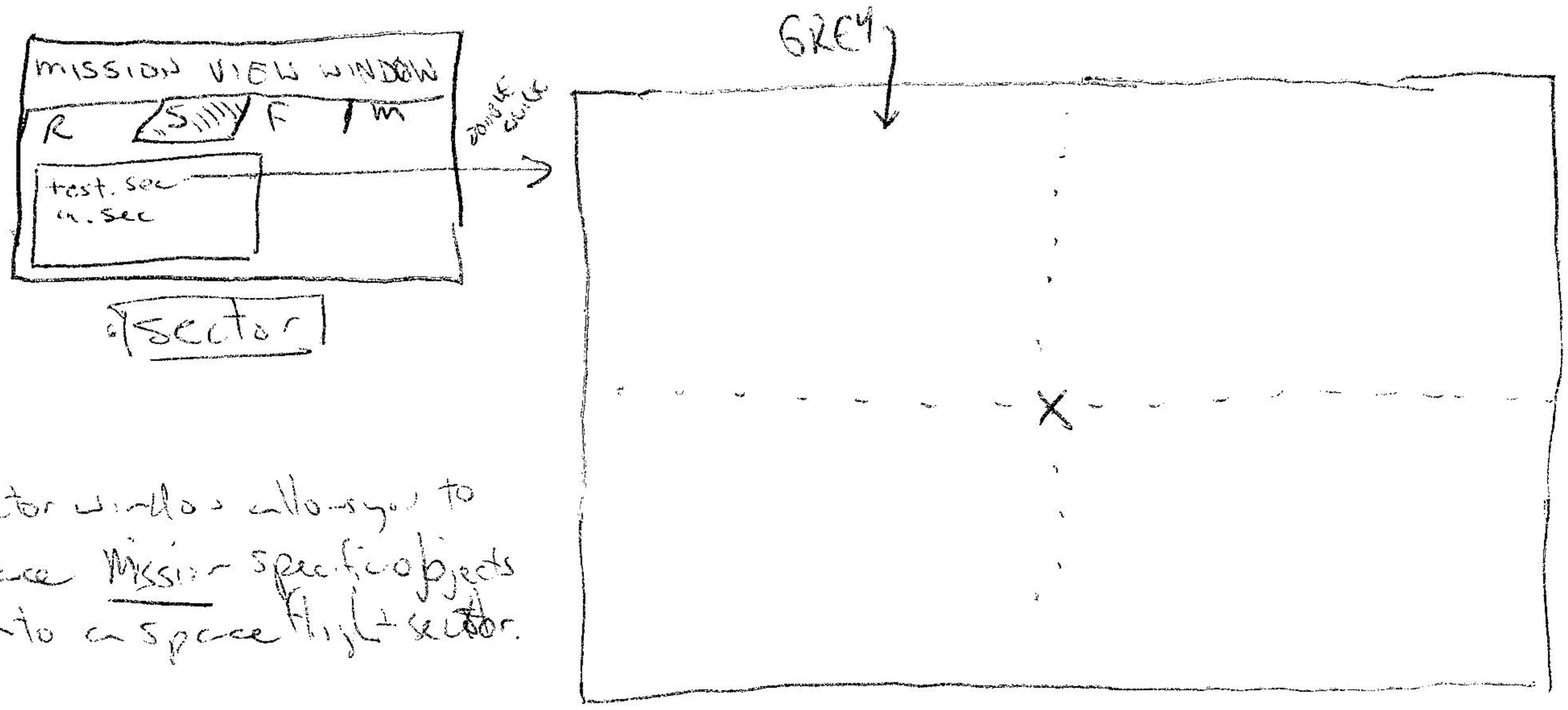
~~void~~ void NAV_SetDescription (SF_String s);
purpose: changes Nav Description in NavMap and NAV VDU to 's'

void NAV_HideNav();
Purpose: Hides Nav in ~~Briefing~~ NavMap

void NAV_ShowNav;
purpose Shows nav in ~~briefing~~ NavMap

void NAV_SetPlayerNav(int new_nav);
purpose: sets the player's next nav to new_nav

MISSION - Control GAMEFLOW + SPACEFLIGHT



Sector windows allow you to place Mission specific objects into a spaceflight sector.

MISC

Properties
- _ObjType_DO_NOT_MODIFY
- FLO_MAIN "FILL THIS IN"
- Start_SEED (RANDOMSEED)
- SAMPLE_LIST (SFX_LIST)

FUNCTION commands (mission specific [MS_*])

void MS_RunGameflow (int start_room_number)

Starts the user in the gameflow room specified by start room number

Start room number starts at 0 + corresponds to the number the room was added to the room list

NOTE: THIS MAY NOT CORRESPOND TO THE ORDER THE ROOMS ARE LISTED.

usage: MS_RunGameflow(0); //starts user in 1st room

void MS_RunSpaceflight (int n); ⟸ n is ignored

- starts the user in the FIRST (and ONLY) sector listed

usage MS_RunSpaceflight (0);

void MS_Run ICIS ();

Runs the ICIS for this mission

usage MS_Run ICIS();

void MS_Run Briefing ( );

Runs the Briefing for the mission

void MS_Exit ( );

exits mission + goes back to series

```
Void    M_MissionMain ()
{
        // see if Vagabond available
        if (!g_VagabondTalkedTo)
        {
            MS_RunGameflow(1);
        }
        else
        {
            MS_Run_Gameflow(0);
        }
        // see if player viewed
        Briefing
        if (g_BriefingViewed)
        {
            MS_RunIcIS ();
        }
        else
            MS_RunBriefing ();
        }

        // Go into Flight
        MS_RunSpaceFlight();
}
```

Cargo Containers

OBJECTS THAT CAN BE IDENTIFIED BY THE PLAYER'S TARGETING SYSTEM

PROPERTIES:

- _Obj_Type_DO_NOT_MODIFY
- Appear Cargo {Shape}
- BriefingDate
- DescCargo
- identifiable — true\false — can be identified Y/N
- identified — true\false — false = UNKNOWN
- Contents — SESTRING — what shows up in target VDU

* FLO_Death
- death program that's run when the object dies
- only one-shot commands are allowed.
- name of a function expected
- if "NULL"

FLO_Main
- Cargo container's main program (See sheet for typical MISSION program commands.

orientation
- y p r in Degrees

* Spawn_number {1 or above}

Start_Nav Point {0 to # of Navs - 1}

* NOT HOOKED UP yet

MED

SF_* commands: "Generic" Spaceflight commands that can be used with almost any type of spaceflight object...

void SF_SetWorldPosition(long x, long y, long z);
allows you to move an object (teleport) to the specified coordinates

ex: // move obj to <-10,000, 2000, 0>
SF_SetWorldPosition(-10,000, 2000, 0);

void SF_ActivateSelf (int display_jump);
: turns the object on - makes the object visible; targetable; collideable; etc

parameter display_jump if Q <u>do not</u> display jump sequence; if non Q = display

void ~~Activate~~ SF_ActivateObject (_ObjId obj, int display_jump)
"Turns object Obj on.

ex: SF_ACTIVATEOBJECT (MarineShuttle, 1);

void SF_DeactivateObject (_ObjId obj, int display_jump):
opposite of

void SF_DeactivateSelf (s int display_jump);
turns object 'off'

ex:

AI_GotoObjRange (JumpPoint, 1000);
SF_DeactivateSelf(1);

void SF_DestroySelf();
Deactivate + Display explosion + Run Death Program

void SF_DestroyObject (_Objid obj);
Deactivate object + Display explosion + Run Death Program

int SF_GetDistance (_Objid obj);
returns distance to obj in meters

ex:

int dist = SF_GetDistance (Player

int SF_Exit();
exits spaceflight

ex:

```
if (SF_GetDistance(Player) < 100)
{
    SF_Exit();
}
```

```
void SF_SetObjectflag (EObjectflags f, int v);

EObjectflags
{
    OF_alignment
    OF_cloaked
    OF_leeched
    OF_vulnerable
}

ALIGN_ALIEN
ALIGN_NEUTRAL
ALIGN_CONFED

ex SF_SetObjectflag(OF_alignment, ALIGN_ALIEN)
   SF_   "          (OF_cloaked, true)
   SF_   "          (OF_vulnerable, 0)  :invulnerable
```

int SF_GetObjectFlag (EObjectFlags f);

ex:

int align = SF_GetObjectFlag (OF_aligned);
OF_vulnerable);

void SF_BindToActionSphere (int new_sphere);
switches the action sphere this object belongs to new_sphere.

void SF_AutopilotEnable (); NOT NEEDED

void SF_PlayerSearchToCam (int cam, _ObjId stunning_target);
SF_PlayerSTC (fro

void SF_SaveObjectState ();
Flag object to be saved from Mission to mission

void SF_SetShipDamage (int main, int child_min, int child_max)
affect a ship w/ main% damage
apply ~~childdam~~ child_min% to child_max%
damage to each child object

SHIP

- ObjType_DO_NOT_MODIFY {...}
- Briefing Data
- Dese Cargo
  - identifiable
  - identified
  - contents
- FLO_Main
  - FLO_Death
  - FLO EventHandler
    - FLO_Event 0
    - -
    - -
    - -
    - Event 7
- Orientation {y, p, r}
- Pilot Comm List
- PilotName
- PilotWingLeader
- Spawn ~~Nav Pt~~ Number {1 ... n}
- ~~Velocity~~ StartNavPoint 0 ... numNavs
- Velocity % of Max no A/B

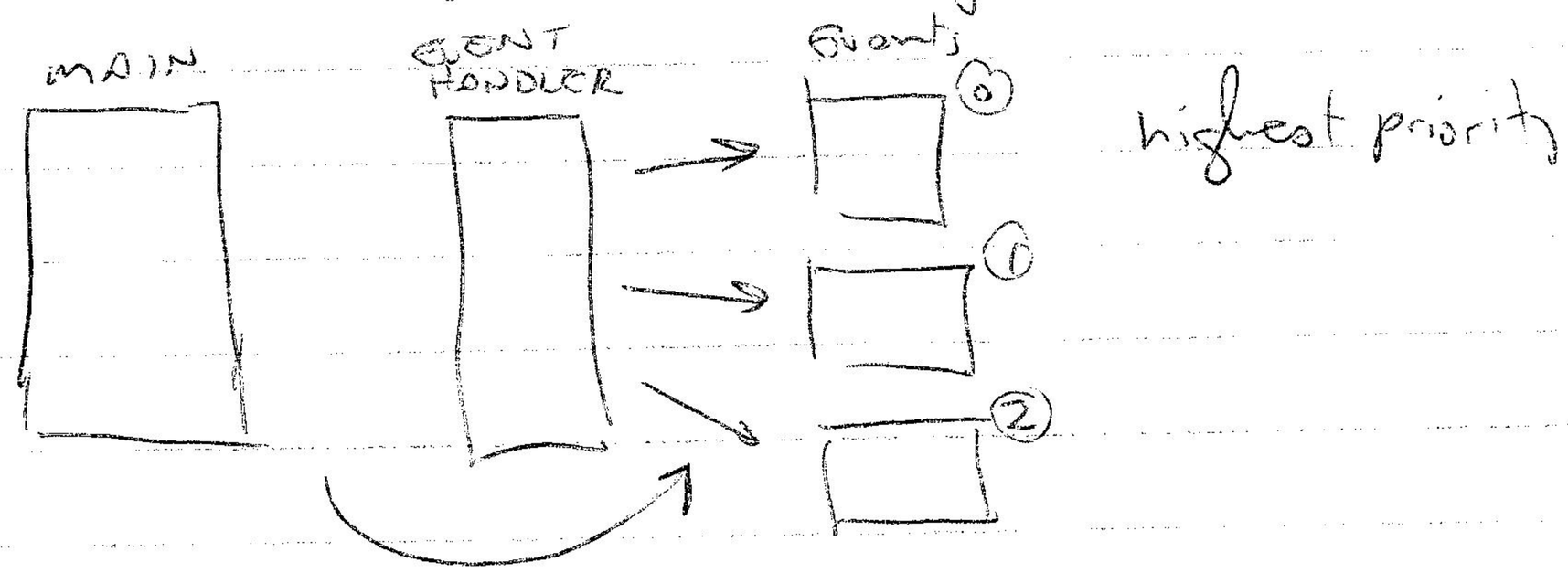
EVENT SYSTEM

The event system allows temporary halting of the currently running command to be interrupted by a new command based on some 'event'

EVENT HANDLER (one shot)

: Program that gets run every "object update" (if enabled)

: EH is responsible for setting events as 'true' or 'false'



NOT ONE SHOT

Events: Programs that can interrupt Main or EVENTS of Lower-Priority

Priority: 0 is highest priority. An ordering scheme for the events.

Event Commands

SYS_EVENT_*

void SYS_EVENT_EnableHandler()
turn on the EH; allow EH processing

void SYS_EVENT_DisableHandler()
turns off the EH;

void SYS_EVENT_Enable(int e);
turn on Event 'e'

void SYS_EVENT_Disable(int e)
turn off Event 'e'

void SYS_EVENT_EnableAll(); turns on all events..
DisableAll(); turns off all events..

void SYS_EVENT_SET(int e)
only used in EH

```
void M_shipMain()
{
    SF_ActivateSelf(1);
    while (1)
    {
        AI_WaitSeconds(1)
    }
}

void M_shipEH()
{
    // check for Xena
    if g forced to Watch
    {
        SYS_EVENT_SET(0)
    }

    // send comm
    if Thresh Killed
    {
        SYS_EVENT_SET(1)
    }

    // Play sound
    if (SF_GETDISTANCE(Player) < ...)
    {
        SYS_EVENT_IS_SET(2)
    }
```

```
void M_EshipSound()
{
    SYS_PLAYSFX(37);
}

void M_EshipComm()
{
    if (!m_commSent)
        "Play_comm"(32)
        SYS_EVENT_DISABLE(x);

void M_fshipSuicide()
{
    SF_DestroySelf();
}
```

- What Does the ship do if not interrupted
  [main program]

- What outside circumstances can influence the ship?
  [Event Handler]

- What do the ships do for each circumstance?
  [Events]

- What is the Priority of these events

M_ShipKiller_E MarinesLanded

When player is near, play sound

IF (g_ThrakhathKilled) is set,
~~commit suicide~~ send comm
IF (g_IntroduceTo Janna)
commit suicide

FUNCTIONS

CURRENT FUNCTIONS:

Predefined

Stand "Outside" Phrases

USER FUNCTIONS:

Unique Source Code (mp4)

Ex:

Vector Math (Angle, ...)

FUNCTION POOL:

LIMITS (POSITIVE USER FUNCTIONS):

1) Not himself does this function actually exist?

2) Sort Length?

a) Does it refer to an existing Function?

Fast Limits

Slow Limits

b) Is it Stable Based Method?

c) Does it Input Parameters?

Convert to ...

void C_yield(Object _Obj1d obj, int radius)

{

int (i)

{

DrawCircle(obj, r, g, b)

}

}

Function that returns the sum of two values

[illegible] that returns the larger of two values

[illegible] of [illegible]

```
int min(int value, int value2)
{
    int num = value2
    if value < value2
    {
        num = value
    }
    return num;
}

int min(int value, int value2)
    int num = value2
    if value < value2
    {
        num = value
    }
    return num;
}

int max(int value, int value2)
{
    int num = value2
    if value > value2
    {
        num = value
    }
    return num;
}
```

```
int fun (int value, int value)
{
    int num;
    num = value + value
    num = num / 2
    return num;
}
```

3 parameters MOZ in ...

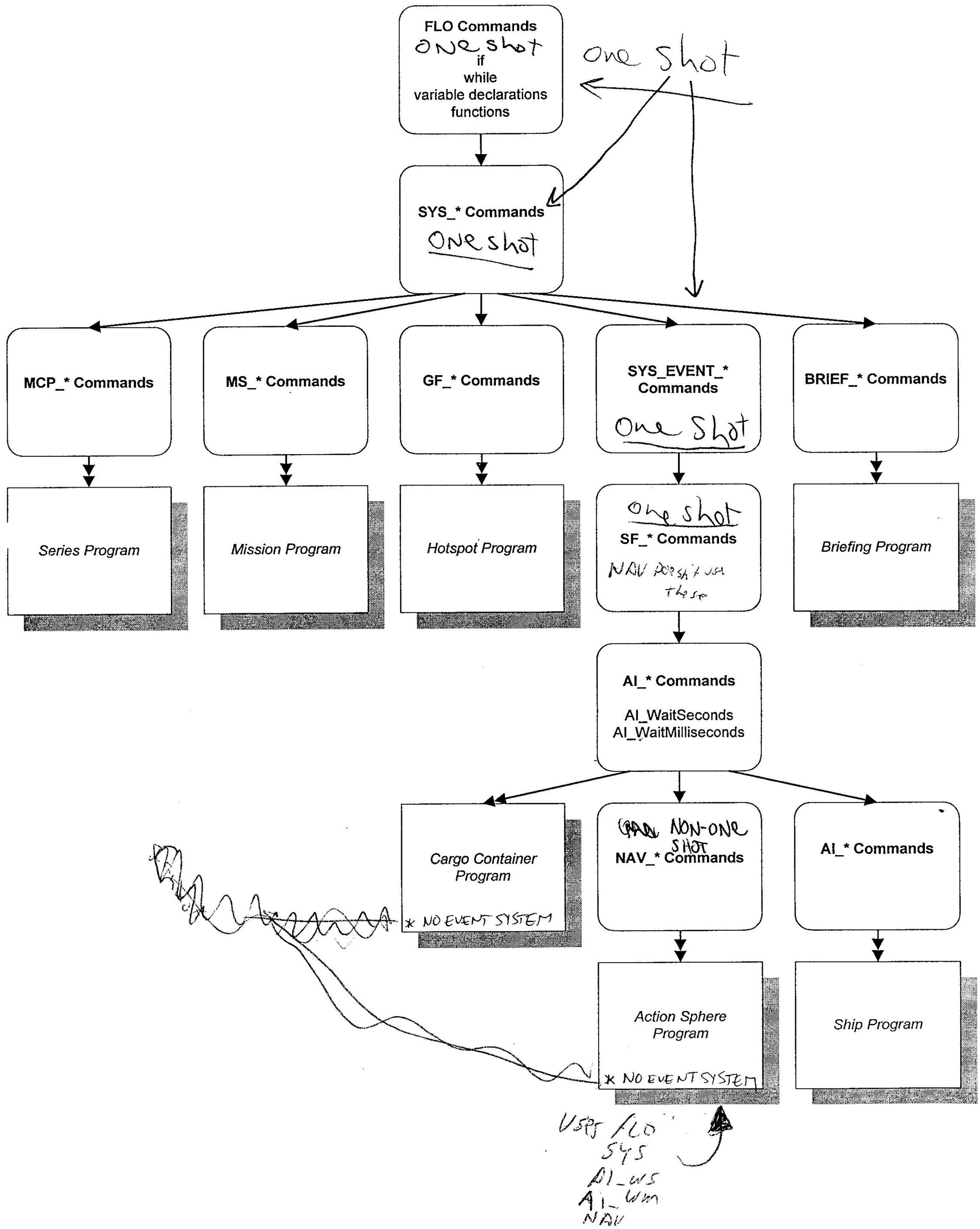
3 local variables Mozilla ...

```
int min3 (int a, int b, int c)
{
    int lowest = a;
    lowest = (min2(a,b));
    lowest = (min2(lowest, c));
    return lowest
}
```

```
int Max3 (int a, int b, int c)
{
    int biggest = 0;

    biggest = (max2(a,b));
    biggest = (max2(biggest, c));

    return biggest

}

int Mean3 (int a, int b, int c)
{
    Return (a+b+c)/3 ;

}
```

Wing Commander : Prophecy

Mission System Program Hierarchy



Cain, Billy

From: Sommers, Andy
Sent: Tuesday, June 10, 1997 8:30 AM
To: ML WCP PROGRAMMERS
Cc: ML WCP DESIGN
Subject: Game Update 1.00.08A

DESIGNERS :

* cockpit=[filename] is back in use - used to load a specific cockpit

* New commands :

```
void MS_ShowTakeoffSequence ();
```

purpose : to show the takeoff sequence
usage : only in mission programs

```
int SF_GetVelocity ();
```

purpose : gets the velocity of the object executing the instruction
usage : any programs that use SF_* commands

```
int velocity = SF_GetVelocity ();
```

```
int SF_GetVelocityObject (_ObjId o);
```

purpose : gets the velocity of the object specified as a parameter
usage : any programs that use SF_* commands

```
int stunning_velocity = SF_GetVelocityObject (Stunning);
```

* Collision now checks the object's collidable flag set with SF_SetObjectFlag in your object's program. Default is collidable.

```
// turn off collision for this object
SF_SetObjectFlag (OF_collidable, false);
```

* Hopefully the crazy crashes when loading missions should be fixed. There were incorrect assumptions made by the cockpit system about the capabilities of certain ships. I addressed the following issues :

- Missile select indicator (CockpitMissileSelect) crashed if the ship had no missiles.
- Gun power indicator (CockpitGunPower) went loony if the ship had no guns.
- Gun select indicator (CockpitGunSelect) left off the parking garage if the ship had no guns.

Sigh... <Shrug>... [Groan]... "Eye-roll"... {Tsk. Tsk.}... `Lament'... (Flail)... hehehe

* You can turn off player collisions with CTRL-C. A message will appear informing you of the current status of the flag.

* You can turn off player invulnerability with CTRL-I. You will still be collided with, but you cannot die. A message will appear informing you of the current status of the flag.

* OF_collidable and OF_vulnerable are now hooked up through SF_SetObjectFlag.

* The movie playing in the game has been disabled (pending the ability to stop the sound once it's started)

* I put asteroids back in...

* The once-disabled plaque system is back in.

— GMP CLASS —

Write good [illegible] for an object

Summaries of [illegible] — NOT SYNOPSIS

How to script [illegible]

For each [illegible]

- ☐ DETERMINE # OF ACTORS/[illegible]
- ☐ DETERMINE # OF OBJECTS

FOR EACH OBJECT [illegible]

OTHERS

- [illegible]
- ↓ [illegible]
- ↓ ☐ WEIGHT

{ [illegible] }

1) IS THIS A [illegible] ?
   VS. [illegible]
2) [illegible]
3) DOES [illegible]
   TO A [illegible]
4) WHAT DO [illegible]
   BECOMES MORE ?
5) [illegible]
   ANGLE ?
6) [illegible]
7) [illegible] HOW THIS OBJECT [illegible]
8) [illegible]
   ANYTHING ?

Becomes inactive

GMP - 1

```
void AI_[illegible]();
void SF_[illegible] Object([illegible]);
void SF_Destroy Object([illegible]);
```

[illegible]

a) [illegible]

b) [illegible]

c) [illegible]

b) can [illegible] other [illegible] "[illegible]"

#5 [illegible]

a) [illegible]

OBJECTS

b) Small [illegible]

c) [illegible]

d) ...

1) 7. Only to [illegible]

[illegible]

(or [illegible])

FRAME RATE

"FRAME [illegible] WINDOW"

Exp. [illegible]

1) ACTUATION

2) [illegible]

3) LIGHTING

i) [illegible]

4) COMPONENTS THAT INVOLVE OBJECTS

[illegible]

ex. SF_Go [illegible];

HELP FOR [illegible]

1) Lighting

a) [illegible]

b) [illegible] 2 directional lights

c) POINT LIGHTS - ONE (PER [illegible])

2) F[illegible]

3) [illegible]

[illegible]

TID [illegible]

1) WORLD [illegible] 4x

[illegible]

THE WORK AE

[illegible]

2) AVG [illegible] (ship 130-180

high Detail Fighter

Capship ~800-1900

5) [illegible]

6 MP [illegible]