Cain, Billy

From: Sommers, Andy
Sent: Monday, March 03, 1997 8:29 PM
To: Cain, Billy; ML WING DESIGN; Hughes, Jason; Roan, Frank
Cc: Shelus, Peter
Subject: RE: Design Issues

As I was putting in the new commands, I remembered that when you double click on the command in MED, it's default parameter is set to 0 (or the first element in the ENUM list). So, for example, when the designers select `SF_ActivateSelf`, MED will automatically use `( FALSE )` as the parameter. This can sort of be used as the "default". It works well in this example :

```
        SF_ActivateSelf ( BOOL display_jump )
```

because it will not display a jump by default. If the designers want to display a jump sequence, they'll use

```
        SF_ActivateSelf ( TRUE )
```

Maybe we should use an enum for this command similar to :

```
        EJumpStatus
        {
                NO_JUMP,
                JUMP
        };
```

which would change the command declaration to :

```
        SF_ActivateSelf ( EJumpStatus display_jump )
```

and the usage to :

```
        SF_ActivateSelf ( NO_JUMP );
```

Just a thought...

-ALS

From: Roan, Frank
Sent: Tuesday, February 25, 1997 3:22 PM
To: Sommers, Andy; Cain, Billy; ML WING DESIGN; Hughes, Jason
Cc: Shelus, Peter
Subject: RE: Design Issues

Any way we can do automatic variables? For example:

definition:
```
SF_ActivateSelf ( int display_jump = FALSE );
```

call:
```
SF_ActivateSelf ();
```

So the designers only have to specify it when they want it? I'm not sure if this is a good idea or not (could make their could confusing), but I've always appreciated it in C++. Just a thought, (fjr)

-----Original Message-----
From: Sommers, Andy
Sent: Tuesday, February 25, 1997 2:57 PM
To: Cain, Billy; ML WING DESIGN; Hughes, Jason
Cc: Roan, Frank; Shelus, Peter
Subject: Design Issues

The following issues were decided upon in our meeting about what was lacking from the game currently :

* need to know when player has fired at object? (mission stats)

* global mission timer display
  action sphere timer display

* give billy control of where to drop the player in the action sphere

* hook nav point to autopilot light

* initial velocity property

The following commands were discussed from the old misison system to be implemented in the new mission system. The probable names of the commands [and parameters] are listed below :

* Need to know if an action sphere is active

```
        SF_IsActive ( _ObjId o );
```

* A command to destroy an object

```
        SF_DestroySelf ();
        SF_DestroyObject ( _ObjId o );
```

* A command to activate an object relative to another object. When this command is used, the orientation specified in MED will be the orientation RELATIVE to the other object.

```
        SF_ActivateObjectRelFrame  ( _ObjId o, long x, long y, long z, int
i, int j, int k );
```

* Need to know when to display a jump effect when an object activates. The following commands will probably change to :

```
        SF_ActivateSelf ( int display_jump );

        SF_ActivateObjectRelFrame ( _ObjId o, long x, long y, long z, int
i, int j, int k, int display_jump );

        SF_DeactivateSelf ( int display_jump );
        SF_DeactivateObject ( _ObjId o, int display_jump );
```

I can't think of a reason where you'd want to display both a jump effect and an explosion, so the display_jump does not apply to SF_Destroy*.

* Need to know ship damage level

```
        SF_GetShipDamage ();
```

* Need to set ship damage

```
        SF_SetShipDamage ( int damage_percent );
```

* Need to play movie!

```
        SF_PlayMovie ( long movie_num );
```

* Need to be able to switch to any camera view

```
        SF_SwitchCamera ( int camera_num );
```

-ALS