

# New Scripts

Many scripts from TDD work in MFP, but some of them have been removed (e.g. SetInventoryDisabled).

However, MFP has a set of its own new scripts, which are listed below.

*This list might be incomplete. You can contribute by going through original MFP .hps files and listing any new scripts ("new" meaning: not a TDD script and not a function which is declared in the same file).*

*You can also contribute by testing the untested functions or function arguments with uncertain usage.*

## Player

```
void SetLanternFlickerActive(bool abActive);
```

Enables the lantern flicker effect.

*abActive* - set to true to enable the effect

```
void SetPlayerInfection(float afAmount);  
void AddPlayerInfection(float afAmount);  
float GetPlayerInfection();
```

Infection related scripts.

## Screen effects

```
void ShowScreenImage(string asImage, int alPosX, int alPosY, float  
afUnknown, bool abUnknown2, float afTime, float afFadeIn, float afFadeOut);
```

Displays an image on the screen. Originally used to show the MFP logo in-game.

*asImage* - the image to display. E.g. startup\_aamfp\_logo.jpg

*alPosX* - horizontal position of the image. 0 is right screen border, smaller values are left. -400 was used for the middle of the screen.

*alPosY* - vertical position of the image. 0 is bottom, smaller values are up. -350 was used for the middle of the screen.

*afUnknown* - Unless set to below 0, the image won't appear. Might have more functionality to it.

*abUnknown2* - setting this to true makes the image not appear. Might have more functionality to it.

*afTime* - image display time.

*afFadeIn* - fade in time. It's added to the base display time.

*afFadeOut* - fade out time. It's added to the base display time.

## Enemies

```
void SetEnemyMoveType(string &in asEnemy, string asMoveType);
```

**? This script hasn't been tested.**

Sets the enemy move type.

*asEnemy* - the in-game enemy entity

*asMoveType* - options include: WalkBiped, RunBiped, ChargeBiped, and probably Quadruped variations as well.

```
void SetManPigType(string &in asEntity, string &in asType);
```

**? This script hasn't been tested.**

Seems to alter the enemy AI. It was mostly used in conjunction with Child enemies (but also with Wretches).

*asEntity* - the enemy in question. Can be Enemy\_ManPig or Enemy\_Child type.

*asType* - only "Freddy" has been used in the entire game. It is unknown whether other options work.

From Peter Howell's PhD paper<sup>1)</sup> :

*"The initial design of the game's enemy artificial intelligence system contained three unique sets of behavioural controls. [...] every enemy agent in the game would be assigned one of three possible 'personalities', referred to in the game's code as the 'Rod', 'Jane' and 'Freddy' personality types."*

*Overview of proposed enemy agent personality types and key behavioural traits (Pre-Development, December 2011):*

### 'Rod':

- Will maintain a 'safe' distance from the player-character.
- If unable to do so, will approach player character, investigate them (by getting close and smelling them), before continuing its patrol.

### 'Jane':

- Will maintain a 'safe' distance from player-character, whilst observing the player-character's movements.
- If unable to maintain 'safe' distance, will panic and flee.
- If cornered and unable to flee, will attack and knock player-character to floor, then flee.
- Will only attack and kill player-character as a last resort.

### 'Freddy'

- Will actively hunt the player-character.
- Will attack and kill them if given the opportunity.

[Source](#)

## Other

```
void SetParticleSystemActive(string &in asParticle, bool abX);
```

“Freezes” a particle (meaning, the last texture will linger).

Manual particle optimising might have been the intended usage. It was seen used next to *SetPhysicsAutoDisable*.

Another apparent usage is to place an inactive particle in the map and activate it when convenient (which might be a simpler way than *CreateParticleSystemAtEntity*)

*asParticle* - the particle in question

*abX* - whether to freeze or unfreeze the particle

```
void AddHint(string &in asEntryName, string asUnknown);
```

Adds a journal entry in the “My Journal” category.

ui\_add\_quest.snt is played, but MFP has only silence in those sounds.

*asEntryName* - entries the lang file.

Like Notes, it must consist of two actual entries: *Hint\_EntryName\_Name* and *Hint\_EntryName\_Text*.

Both must be in the “Journal” category.

See english.lang for examples.

*asUnknown* - this argument was never used. It always contained an empty string. Writing in it doesn't yield any error messages or log entries.

```
SetLightVisible(string& asLightName, bool abVisible);
```

**⚠️ This script hasn't been tested.**

Enables/disables lights.

It was used in the main game scripts (in conjunction with SpotLights), which might indicate that unlike in TDD, this time it actually works.

*asLightName* - internal name

*abVisible* - determines the state of the light

```
void SetPhysicsAutoDisable(string &in asEntity, bool abX);
```

**⚠️ This script hasn't been tested.**

Unknown usage.

*asEntity* - entity in question. It was used with chandelier\_nice.

abX - whether to disable the object physics (*when?*).

<sup>1)</sup> "Disruptive Game Design: A Commercial Design and Development Methodology for Supporting Player Cognitive Engagement in Digital Games", Peter Howell, 2015

From:

<https://wiki.frictionalgames.com/> - **Frictional Game Wiki**

Permanent link:

[https://wiki.frictionalgames.com/hpl2/machine\\_for\\_pigs/new\\_scripts?rev=1583689773](https://wiki.frictionalgames.com/hpl2/machine_for_pigs/new_scripts?rev=1583689773)

Last update: **2020/03/08 17:49**

