The Netcrawl Protocol

0. Introduction

The Netcrawl Protocol describes and outlines the eponymous game, which at its core is an abstract dungeon crawler, battler and puzzle game. It is by all means an open, free, adaptable and public domain game.

1. Definition

A game of Netcrawl occurs in an universe structured as interlinked “world” cells, each containing their own “node” cells. The method through which “world” and “node” cells are interlinked and traveled to and from is not set.

Each “world” cell typically starts in “lockdown mode”, preventing entities within from traveling to other “world” cells. The method through which “lockdown mode” is revoked is not set.

Entities inhabit “node” cells, and are composed of:

- -an identifier (usually a string)
- -an allegiance component (specifics not set)
- -resource values / ports (outlined later)

All entities have the same basic abilities, outlined in the command line functions. The amount of actions relative to time each entity can carry out is not defined.

Entity resources / ports are composed of:

- -a polarity (send / request, or in/out, or R/S, etc)
- -a magnitude value (in the 1 → 16 range in Netcrawl I)
- -a resource amount value (starts at 255 in Netcrawl I)

Entities mainly challenge each other through trading, which they can primarily do by populating the same Node Cell.

All entities in a node cell will trade with each other simultaneously. For this reason, there are two stages to trading: damage accumulation and damage resolution.

Entities A and B Trade() simultaneously in the following way:

-Entity A's send ports damage entity B's identical send ports, multiplied by entity A's individual port magnitudes and disfavor towards entity B.

-Entity A's send ports damage entity B's identical request ports, multiplied by entity B's individual port magnitudes and A's disfavor towards entity B.

-Entity A's send ports heal entity B's identical request ports, multiplied by entity B's individual port magnitudes and A's favor towards entity B.

-Entity B behaves the same towards entity A, simultaneously.

-Request ports remain passive.

Pseudocode example: (A trades with B)

```
for(int x = 0; x < global_maxports; x++){
        int favor; //calculated off allegiance of entities A and B. not always symmetrical
        int disfavor;
        int damage_to_A;
        int damage_to_B;
        if(entity_A.ports[x].polarity == “send” && entity_B.ports[x].polarity == “send”){
                //”damage” dealt hinges on sender magnitude and disfavor
                damage_to_B= entity_A.ports[x].magnitude * disfavor;
                damage_to_A= - entity_A.ports[x].magnitude * disfavor;
        }
        if(entity_A.ports[x].polarity == “send” && entity_B.ports[x].polarity == “receive”){
                //”damage” dealt hinges on sender disfavor and receivor magnitude
                damage_to_B= entity_B.ports[x].magnitude * disfavor;
                damage_to_A= - entity_B.ports[x].magnitude * disfavor;
                //”sending” to a “favored” receiver will “heal” it, at your expense
                damage_to_B = - entity_B.ports[x].magnitude * favor;
                damage_to_A= entity_B.ports[x].magnitude * favor;
        }
        entity_A.ports[x].TakeDamage(damage_to_A);
        entity_B.ports[x].TakeDamage(damage_to_B);
}
```

-Ports whose resource value fall negative receive “damage”. How damage is resolved is not specific.

-An entity which receives critical damage is destroyed.

-Damaged ports can be employed to modify an entity’s allegiance.

-When a majority allegiance profile populates a Node Cell, the Node Cell in question inherits said allegiance.

-Majority rule is (by default) decided by total resource points, rather than total discrete entities.

-When trading, entities also take into account the other’s allegiance profile. The “default” method mitigates damage dealt per tokens in common, and accentuates damage dealt per tokens not found in the adversary.

Example: entity A ([1], [4], [7], [9]) will greatly damage entity B([4], [5]) because B lacks tokens [1], [7], [9]. B slightly damages A because it shares token [4], and only lacks token [5].

-Accumulating allegiance tokens is generally a good idea, although revoking tokens is sometimes necessary for the more specific designed interactions.

-The same principle extends to World Cells.

-The “Player Entity” achieves victory (dubbed “Netcrawl”) once every World Cell is turned to its allegiance.

2. Basic command definitions

NOTE: a game of Netcrawl roughly implements the following commands, as well as any number of other unspecified commands. A game of Netcrawl does not necessarily document its available commands.

“netcrawl” begins the game, and ends the game if win conditions are met

“help” may or may not display help

“Show” [target] will display information on given entities matching the identifier [target]. What information netcrawl entities have access to through this is not specified.

“Move” [target] moves its caster to any given location matching the identifier [target], provided that movement is allowed.

“Trade” [target] initiates one instance of trading with named target.

“Sleep” [amount] disables its caster for the specified time span.

“Alias” [alias] [command] allows you to create custom macros and aliases for commands and command sequences. It is a quality of life command.

“Set” [data] allows an entity to set its identifier, and port settings. In some cases, it will allow variable declarations when executing gracefully. Variables should function as aliases.

“Use” [name] [etc] uses any item/nondescript according to its own abilities

3. Additional Information

Entities may also possess items of any nature and ability. Common items include resource banks, entries and logs, special weaponry (such as those allowing casters to bypass trading) etc.

Every implementation of the Netcrawl protocol should be different from others. Players who attempt an iteration of Netcrawl for the first time should always be expected to have a learning phase and an experimentation phase before ever possibly achieving a Netcrawl.

The command system does not have to be represented in a command line form. Overall, a game of Netcrawl does not have a set form of representation.

A game of Netcrawl will typically generate with certain components randomized. For instance, Netcrawl I’s port names, faction allegiances, world connections and node counts are randomized. Furthermore, certain rules for damage calculation and allegiance value are also randomized, and must be understood through one of a few conceivable methods.

Because all entities, player and not, share the same basis, a game of Netcrawl has no set number of human and machine players required.