RTOR: Roguelike Theory of Relativity

(Fully Turn-based Multi-player)

1. Introduction

Roguelike games are often defined by a set of characteristics. This usually includes "single player" as a key defining characteristic. This allows a player to stop and think about a tricky situation, and the alternatives they have to live. As such a large number of roguelike games are single player, including Nethack, Angbang etc - retaining the "strategy" aspect of the games.

While multi player variations of roguelike games do exist, they are typically based on real-time movement. Examples include Crossfire and MAngbang. Due to the real-time nature of these games, certain liberties are taken to "automatically" control the game such as "automatically" attacking monsters near to the player. This effectively reduces the role of the player, and the need for "strategy".

The Roguelike Theory of Relativity (RTOR) is based on maintaining the player-centric gameplay of roguelike games, while allowing for multiple players to fairly interact. In other words, fully turn-based multi-player roguelike gaming.

2. RTOR and Concept Development

Under RTOR, everything in the game is player centric (everything revolves around player movement). In a single-player model, when a player moves, a monster can move as well (pending speed differences etc). This also simplifies all game loops as well, as each time a player moves, we simply check every monster on the level to see if they can move as well.

By introducing other players to a level, this player centric model remains in place, and monsters are "allocated" to players based on various scenarios and number of turns taken by players.

RTOR originally started as "Sureal Time", a theory where players would enter a real time movement sequence when in proximity to each other. Thus players that travel the dungeon alone will always operate in a single player / turn based move. Players in close proximity would operate in a real time mode - however if no players moved within a certain time, nothing would happen.

Sureal Time was a good idea, but overly complex to implement. And given one of the core considerations of roguelike games is "turn based" movement allowing strategic thinking, Sureal Time was abandoned.

3. Scenario: Single Player

In this scenario, a single player (@) and monster (M) are the only active creatures on a level. Every time the player moves, the monster may move (dependent on speed of each monster).

```
----------
|........|
|.M.@.....
|........|
----------
```

This movement algorithm is how a normal roguelike game operates. As such no changes are required: A multi-player roguelike applying the RTOR will invariably behave like a single player game when only one player is on a level.

4. Scenario: Player on Player

The next scenario considers player A and player B attacking each other:

```
----------
|........|
|...AB....
|........|
----------
```

When player A attacks player B, a record is maintained by player A that they attacked player B. This record contains information on player A's speed. If player A attempts to attack player B a second time, the record is checked, and as there is an imbalance in the record, the attack by player A fails (no movement penalty).

When player B attacks player A, two things will occur. Firstly the record maintained of the initial attack by player A is modified in accordance with player speeds. Secondly player B then creates and maintains a record of hitting player A. Thus player B can no longer attack player A, but because the turn based speed has been applied, player A can now retaliate.

Comment	Player A	Player B
Speed of players:	12 (normal)	8 (faster)
Player A hits B, attack success as record for B initially set to 0 < 8	B = 0 B = 0 + 12	
Player A tries to hit B, but as B >= 8, attack fails	B = 12	
Player B hits A, attack success as record for A initially set to 0 < 12	B = 12 - 8	A = 0 A = 0 + 8
* Player B hits A, attack success as record for A < 12	B = 4 - 8	A = 8 A = 8 + 8
Player B hits A, attack fails as record for A >= 12	B = -4	A= 16
Player A tries to hit B, attack success as record for B < 8	B = -4 B = -4 + 12	A = 16-12
Player A tries to hit B, attack fails as record for B >= 8	B = 8	A = 4
Player B hits player A, attack success as record for A < 12		A = 4 + 8
Final speed records	B = 0	A = 12

The table above shows an example of how two players with differing speeds would interact. The move marked with an asterix (*) is where player B has “gained” a move as a result of being a faster player - turn wise (NOT connection speed!).

This implementation means regardless of connection speeds, player-on-player interaction is based on turn based movement speeds. It also means it is up to the player if they wish to attack other players. If a player decides not to attack any players, they may only occasionally need to tolerate one or two attacks by another player.

This method is effectively a more complicated method used between each monster and player. In the case of monster-player interaction, the monster simply carries one record for the player which they are tied to. Because there are multiple players to track, each player needs to maintain a “state” for other players they have encountered within the game. Each player must maintain a separate state in order to ensure a record is kept of the behavior (conduct) of the player.

5. Scenario: Multiple players

When more than one player exists on a level, each monster is tied to a particular player. This will typically be the player closest to the monster. In the example below, the monster (M) is tied to the player (@) at the bottom of the room. Every time a player moves, any monsters tied to that player also move (in accordance with appropriate monster speeds).

Every time ANOTHER player moves, a counter is incremented for the monster. This counter is reset every time the player the monster is tied to also moves. Thus if a monster is tied to a player, they remain tied to that player so long as movement continues.

```
-----------
|...321...|
|.M....@...
|...@.....|
-----------
```

If the counter reaches a certain value (i.e. the player they are tied to has stopped moving for some period of time), the monster is reallocated to a moving player. Thus if the player at the bottom of the room stops moving, and the player at the side of the room moves as per (1, 2, 3), the monster M will be tied to the moving player at about when the player reaches point '3'.

In order to reduce monsters “churning” between two players in close proximity, the switch to another player is exponential. Thus a monster 5 spaces from a player is more likely to switch to another player than a monster that is 2 spaces from the same player. This also creates a better game balance when larger numbers of players and monsters are on a level.

As a general rule, if two players are attacking a monster, the monster can respond to each player in each respective turn. Thus if player A hits, the monster will be tied to player A, and given the opportunity to respond. Then when player B hits, the monster is tied to player B, and also given the opportunity to respond. While this is not entirely accurate (as the monster would need to adjust attacking direction etc), it is far simpler to code than dealing with changes in direction of attack.

These rules also reduce the benefit of stat grinding. If player A stands still with monsters all around them, and player B tries to stat grind, the monsters will in due process become tied to player B the longer they try and stat grind.

6. Scenario: Player Blocking Monster

There is however one remaining issue, and this is in the following situation:

```
#######M@###@#####
```

In this case, the player next to the monster M remains stationary. The other player can move about and regain HP etc while the monster is blocked from coming down the corridor. This is an effective method of stat grinding under RTOR.

There are very few “believable” options to use in this case. One obvious solution is to allow the monster to “push past” the player they are not tied to. We could swap the monster and player's positions - and thus maintain the one monster per location rule.

7. Conclusion

For years the assumption has been traditional roguelike games can not be multi-player and turn-based. This represented a contradiction to a core concept of roguelike games.

RTOR implements a player centric concept where each monster is tied to a player. So long as players are moving, monsters will then take a turn to move - or if they player they are tied to stops moving, they may switch to another player.

Various considerations are required in order to ensure fair gameplay (particularly between two players), and prevent monsters from switching in certain cases.

Interhack Versions 2.1.0 and above implement this concept as a solution.