

# Role Playing DB V2.0 – Tcl Interface

## Contents

<b>1</b>	<b>Copyright Info</b>	<b>2</b>
<b>2</b>	<b>Introduction</b>	<b>2</b>
<b>3</b>	<b>Dice Class</b>	<b>3</b>
3.1	class Dice . . . . .	3
<b>4</b>	<b>Record Class</b>	<b>3</b>
4.1	class Record . . . . .	4
<b>5</b>	<b>Character Class</b>	<b>4</b>
5.1	class Character . . . . .	7
<b>6</b>	<b>Monster Class</b>	<b>11</b>
6.1	class Monster . . . . .	13
6.2	class MonsterInstance . . . . .	16
<b>7</b>	<b>Spell Classes</b>	<b>17</b>
7.1	class Spell . . . . .	18
<b>8</b>	<b>Dressings Classes</b>	<b>20</b>
8.1	Treasure Class . . . . .	20
8.1.1	class Treasure . . . . .	21
8.2	TrickTrap Class . . . . .	24
8.2.1	class TrickTrap . . . . .	24
8.3	Dressing Class . . . . .	25
8.3.1	class Dressing . . . . .	26
<b>9</b>	<b>Space classes</b>	<b>27</b>
9.1	Geometric constants. . . . .	27
9.1.1	class GeoConstants . . . . .	27
9.2	class Exit . . . . .	28
9.2.1	class Exit . . . . .	28
9.3	ExitVector Class . . . . .	29
9.3.1	class ExitVector . . . . .	29

9.4	Item Class . . . . .	30
9.4.1	class Item . . . . .	30
9.5	ItemVector Class . . . . .	31
9.5.1	class ItemVector . . . . .	31
9.6	Space Class . . . . .	32
9.6.1	class Space . . . . .	32
<b>References</b>		<b>36</b>
<b>Index</b>		<b>37</b>

## 1 Copyright Info

Role Playing DB – A database package that creates and maintains a database of RPG characters, monsters, treasures, spells, and playing environments.

Copyright (C) 1995,1998-1999 Robert Heller D/B/A Deepwoods Software  
51 Locke Hill Road  
Wendell, MA 01379-9728

This program is free software; you can redistribute it and/or modify it under the terms of the GNU General Public License as published by the Free Software Foundation; either version 2 of the License, or (at your option) any later version.

This program is distributed in the hope that it will be useful, but WITHOUT ANY WARRANTY; without even the implied warranty of MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE. See the GNU General Public License for more details.

You should have received a copy of the GNU General Public License along with this program; if not, write to the Free Software Foundation, Inc., 675 Mass Ave, Cambridge, MA 02139, USA.

## 2 Introduction

This manual describes the Tcl interface to the C++ class library used by the Role Playing DB. This consists of a collection of classes for the various base data structures used to represent the various informational objects used in Role Playing Games. These informational objects consist of *Characters*, *Monsters*, *“Magic” Spells*, *(Dungeon) Dressings*, and *“Spaces”*. All of these data objects are built upon a common low-level data object, a **Record**. Also included is a specialized random number generator that emulates one or more dice, dice being commonly used to implement

random **chance** or “fate” in most Role Playing Games<sup>1</sup>.

Although TSR's *Advanced Dungeons & Dragons* [1], [2], and [3] were used as a reference in the design of these data structures, they should be generic enough to be usable for any other Role Playing Game system.

## 3 Dice Class

Dice Class – Just a random number generator using a dice model.

### 3.1 class Dice

[ created from class Dice ]

**Dice name ?ns? ?nd?**

[ Constructor: returns Dice \* ]  
 Constructor, create 1 or more dice.

**rename obj**

[ Destructor: returns void ]  
 Destructor

**Roll**

[ Member : returns unsigned int ]  
 Roll them bones...

**TypeOfDice**

[ Member : returns bool ]  
 Return the type of dice.

## 4 Record Class

Record structure.

Core resident record - it has a size and a chunk of memory (the record itself). This structure is lifted from the Home Librarian package. A very re-usable piece of code.

---

<sup>1</sup>Some use shuffled decks of cards and others use a mix.

Lots of fun. I wish C++ had a garbage collector...

#### 4.1 class Record

[ created from struct Record ]

##### Record name

[ Constructor: returns Record \* ]

Constructor: preallocated buffer.

##### rename obj

[ Destructor: returns void ]

Destructor: free up allocated memory.

##### ReturnRecord

[ Member : returns TclRecord \* ]

Return the Record in printable form.

##### SetRecord source

[ Member : returns void ]

Set the Record from its printable form.

##### ReadRecord filename

[ Member : returns int ]

Load Record from a file.

##### WriteRecord filename

[ Member : returns int ]

Write Record to file.

## 5 Character Class

Basic character class. Contains all of the information needed to describe a player or non player character.

**NewCharacter args**

```
[ Native method : NewCharacter ]
```

This is the interface to the constructor for **Character** objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the **Character** function with a **-this** argument to do that: `[Character -this [NewCharacter options...]]`.

The defined options for this function are:

flag	type	default	explanation
-strength	integer	0	Set the initial character strength value
-intelligence	integer	0	Set the initial character intelligence value
-wisdom	integer	0	Set the initial character wisdom value
-dexterity	integer	0	Set the initial character dexterity value
-constitution	integer	0	Set the initial character constitution value
-charisma	integer	0	Set the initial character charisma value
-exceptionalstrength	integer	0	Set the initial character exceptional strength value
-level	integer	1	Set the initial character level
-hitdice	integer	6	Set the initial character hit die type (number of sides)
-numhitdice	integer	1	Set the initial character number of hit dice
-maxhitdice	integer	1	Set the initial character maximum number of hit dice
-name	string	""	Set the initial character name
-player	string	""	Set the initial character player name
-race	string	"Human"	Set the initial character race
-characterclass	string	"npc"	Set the initial character character class
-alignment	string	""	Set the initial character alignment
-sex	string	""	Set the initial character sex
-age	integer	0	Set the initial character age
-commentary	string	""	Set the initial character commentary
-image	string	""	Set the initial character image (GIF file)

**5.1 class Character**

[ created from class Character ]

**Character name**

[ Constructor: returns Character \* ]  
Base Constructor.

**rename obj**

[ Destructor: returns void ]  
Destructor: free up allocated memory.

**HitPoints**

[ Member : returns int ]  
Return hit points.

**Age**

[ Member : returns int ]  
Return age.

**Strength**

[ Member : returns int ]  
Return strength.

**Intelligence**

[ Member : returns int ]  
Return intelligence.

**Wisdom**

[ Member : returns int ]  
Return Wisdom.

**Dexterity**

[ Member : returns int ]  
Return dexterity.

**Constitution**

```
[ Member : returns int ]  
Return constitution
```

**Charisma**

```
[ Member : returns int ]  
Return charisma.
```

**ExceptionalStrength**

```
[ Member : returns int ]  
Return exceptional strength.
```

**Level**

```
[ Member : returns int ]  
Return level.
```

**ExperiencePoints**

```
[ Member : returns int ]  
Return experience points.
```

**Gold**

```
[ Member : returns int ]  
Return gold.
```

**Name**

```
[ Member : returns char * ]  
Return name.
```

**Player**

```
[ Member : returns char * ]  
Return player.
```

**Race**

```
[ Member : returns char * ]  
Return race.
```

**CharacterClass**

[ Member : returns char \* ]  
Return character class.

**Alignment**

[ Member : returns char \* ]  
Return alignment.

**Sex**

[ Member : returns char \* ]  
Return sex.

**Comments**

[ Member : returns char \* ]  
Return comments.

**Image**

[ Member : returns char \* ]  
Return image.

**SetAge newage**

[ Member : returns int ]  
Set age.

**SetStrength newS**

[ Member : returns int ]  
Set strength.

**SetIntelligence newI**

[ Member : returns int ]  
Set intelligence.

**SetWisdom newW**

[ Member : returns int ]  
Set wisdom.

**SetDexterity newD**

[ Member : returns int ]  
Set dexterity.

**SetConstitution newC**

[ Member : returns int ]  
Set constitution.

**SetCharisma newCh**

[ Member : returns int ]  
Set charisma.

**SetExceptionalStrength newES**

[ Member : returns int ]  
Set exceptional strength.

**SetExperiencePoints newEP**

[ Member : returns int ]  
Set experience points.

**SetGold newGP**

[ Member : returns int ]  
Set gold.

**SetName newN**

[ Member : returns char \* ]  
Set name.

**SetPlayer newP**

[ Member : returns char \* ]  
Set player.

**SetRace newR**

[ Member : returns char \* ]  
Set race.

**SetCharacterClass newCh**

[ Member : returns char \* ]  
Set character class.

#### **SetAlignment newA**

[ Member : returns char \* ]  
Set alignment.

#### **SetSex newS**

[ Member : returns char \* ]  
Set sex.

#### **SetComments newC**

[ Member : returns char \* ]  
Set comments.

#### **SetImage newI**

[ Member : returns char \* ]  
Set image.

#### **AdvanceLevel**

[ Member : returns void ]  
Advance character's level.

#### **UpdateFromRecord rec**

[ Member : returns void ]  
Update character from Record.

#### **RawData**

[ Member : returns Record \* ]  
Type conversion: convert to a Record.

## **6 Monster Class**

Basic monster class. Contains all of the information needed to describe a monster.

**NewMonster args**

[ Native method : NewMonster ]

This is the interface to the constructor for **Monster** objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the **Monster** function with a **-this** argument to do that: [**Monster -this [NewMonster options...]**]. The defined options for this function are:

flag	type	default	explanation
-hitpoints	integer	0	The monster's hitpoints (only if -hittype is Monster::Points).
-hitdie	integer	8	The monster's hit die (only if -hittype is Monster::Dice).
-numhitdice	integer	1	The number of hit dice (only if -hittype is Monster::Dice).
-hitadjust	integer	0	The hit adjustment (only if -hittype is Monster::Dice).
-armorclass	integer	0	The monster's armor class
-landspeed	integer	0	The monster's land speed
-flyingspeed	integer	0	The monster's flying speed
-swimmingspeed	integer	0	The monster's swimming speed
-burrowingspeed	integer	0	The monster's burrowing speed
-webspeed	integer	0	The monster's web speed
-percentlair	integer	0	The monster's percent in lair
-numattacks	integer	0	The monster's number of attacks
-mindamage	integer	0	The monster's minimum damage per attack
-maxdamage	integer	0	The monster's maximum damage per attack
-magicres	integer	0	The monster's magical resistance
-minnumappear	integer	0	The minimum number appearing
-maxnumappear	integer	0	The maximum number appearing
-intelligence	enum	-	The monster's intelligence rating
-frequency	enum	-	The monster's frequency of occurrence

flag	type	default	explanation
-hitttype	enum	-	The monster's hit type
-size	double	0.0	The monster's size
-name	string	""	The name or type of monster
-alignment	string	""	The monster's alignment
-treasure	string	""	The monster's treasure type
-specialatt	string	""	The monster's special attack modes
-specialdef	string	""	The monster's special defensive modes
-psionics	string	""	The monster's psionic abilities
-commentary	string	""	Additional commentary
-image	string	""	Name of image file

## 6.1 class *Monster*

[ created from class *Monster* ]

### Monster name

[ Constructor: returns *Monster* \* ]  
Base Constructor.

### rename obj

[ Destructor: returns void ]  
Destructor.

### Name

[ Member : returns char \* ]  
Return Name/Type.

### Alignment

[ Member : returns char \* ]  
Return the alignment.

### TreasureType

[ Member : returns char \* ]  
Return the treasure type.

**SpecialAttacks**

[ Member : returns char \* ]

Return the special attacks.

**SpecialDefences**

[ Member : returns char \* ]

Return the special defenses.

**Psionics**

[ Member : returns char \* ]

Return the psionics.

**Comments**

[ Member : returns char \* ]

Return the commentary.

**Image**

[ Member : returns char \* ]

Return the image.

**HitDieSides**

[ Member : returns int ]

Return the hit die sides.

**NumHitDice**

[ Member : returns int ]

Return the number of hit dice.

**HitAdjustment**

[ Member : returns int ]

Return the hit dice adjustment.

**ArmorClass**

[ Member : returns int ]

Return the armor class.

**LandSpeed**

```
[ Member : returns int ]  
Return the speed on land.
```

**FlyingSpeed**

```
[ Member : returns int ]  
Return the speed in the air (flying).
```

**SwimmingSpeed**

```
[ Member : returns int ]  
Return the speed in the water (swimming).
```

**BurrowingSpeed**

```
[ Member : returns int ]  
Return the speed in the earth (burrowing).
```

**WebSpeed**

```
[ Member : returns int ]  
Return the speed in web.
```

**PercentInLair**

```
[ Member : returns int ]  
Return the percent in lair.
```

**NumberOfAttacks**

```
[ Member : returns int ]  
Return the number of attacks.
```

**MagicalResistance**

```
[ Member : returns int ]  
Return the magical resistance.
```

**HitPoints**

```
[ Member : returns int ]  
Return the number of hit points.
```

**DamagePerAttack**

[ Member : returns void ]  
Return the damage per attack.

**NumberAppearing**

[ Member : returns void ]  
Return the number appearing.

**Size**

[ Member : returns double ]  
Return the size.

**UpdateFromRecord rec**

[ Member : returns void ]  
Update monster from Record.

**RawData**

[ Member : returns Record \* ]  
Type conversion: convert to a Record.

**Intelligence**

[ Member : returns char \* ]  
Return the intelligence rating.

**Frequency**

[ Member : returns char \* ]  
Return the frequency.

**Hittype**

[ Member : returns char \* ]  
Return the hit type.

Monster “instance”. Used to implement a working instance of a monster.

**6.2 class MonsterInstance**

[ created from class MonsterInstance ]

**MonsterInstance name iOf ?iN?**

[ Constructor: returns MonsterInstance \* ]  
 Constructor.

**rename obj**

[ Destructor: returns void ]  
 Destructor.

**InstanceName**

[ Member : returns char \* ]  
 Return the name.

**InstanceHitPoints**

[ Member : returns int ]  
 Return the hit points.

**UpdateInstanceHitPoints adj**

[ Member : returns int ]  
 Adjust the hit points.

**InstanceOf**

[ Member : returns Monster \* ]  
 Return base monster object.

## 7 Spell Classes

Basic spell class. Contains all of the information needed to describe a spell.

**NewSpell args**

[ Native method : NewSpell ]

This is the interface to the constructor for **Spell** objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the **Spell** function with a **-this** argument to do that: `[Spell -this [NewSpell options...]]`.

The defined options for this function are:

flag	type	default	explanation
-class	string	""	Character class that can use the spell.
-name	string	""	Name of the spell.
-type	string	""	Type of spell.
-description	string	""	Description of spell.
-area	string	""	Area effected by spell.
-casttime	string	""	Casting time.
-duration	string	""	Duration of spell.
-savethrow	string	""	Saving throw (if any).
-level	integer	0	Spell level.
-range	integer	0	Spell range.
-reversible	boolean	false	Spell reversibility.
-verbal	boolean	false	Spell has verbal component.
-somatic	boolean	false	Spell has somatic component.
-material	boolean	false	Spell has material component.

## 7.1 class Spell

[ created from class Spell ]

### Spell name

[ Constructor: returns Spell \* ]  
Base constructor.

### rename obj

[ Destructor: returns void ]  
Destructor.

### UpdateFromRecord rec

[ Member : returns void ]  
Update Spell from Record.

### SpellClass

[ Member : returns char \* ]  
Return the spell's class.

**Name**

[ Member : returns char \* ]  
Return the spell's name.

**SpellType**

[ Member : returns char \* ]  
Return the spell's type.

**Description**

[ Member : returns char \* ]  
Return the spell's description.

**AreaOfEffect**

[ Member : returns char \* ]  
Return the spell's area of effect.

**CastingTime**

[ Member : returns char \* ]  
Return the spell's casting time.

**Duration**

[ Member : returns char \* ]  
Return the spell's Duration.

**SavingThrow**

[ Member : returns char \* ]  
Return the spell's saving Throw.

**Level**

[ Member : returns int ]  
Return the spell's level.

**Range**

[ Member : returns int ]  
Return the spell's range.

**ReversibleP**

```
[ Member : returns bool ]
Return spell's reversibility flag.
```

**VerbalP**

```
[ Member : returns bool ]
Return spell's verbal component flag.
```

**SomaticP**

```
[ Member : returns bool ]
Return spell's somatic component flag.
```

**MaterialP**

```
[ Member : returns bool ]
Return spell's material component flag.
```

**RawData**

```
[ Member : returns Record * ]
Type conversion: convert to a Record.
```

## 8 Dressings Classes

Various dressing classes.

### 8.1 Treasure Class

Treasures are the things the characters are trying to get and the monsters are guarding.

**NewTreasure args**

```
[ Native method : NewTreasure ]
```

This is the interface to the constructor for **Treasure** objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the **Treasure** function with a **-this** argument to do that: `[Treasure -this [NewTreasure options...]]`.

The defined options for this function are:

flag	type	default	explanation
-name	string	""	Name of the treasure.
-description	string	""	Description of treasure.
-image	string	""	Image file (GIF) of treasure.
-weight	integer	0	Weight of treasure.
-armorclassadj	integer	0	Armor class adjustment.
-tohitadj	integer	0	To hit adjustment.
-damageadj	integer	0	Damage adjustment.
-magicalresistanceadj	integer	0	Magical resistance adjustment.
-damageprotectionadj	integer	0	Damage protection adjustment.
-strengthadj	integer	0	Strength adjustment.
-intelligenceadj	integer	0	Intelligence adjustment.
-wisdomadj	integer	0	Wisdom adjustment.
-dexterityadj	integer	0	Dexterity adjustment.
-constitutionadj	integer	0	Constitution adjustment.
-charismaadj	integer	0	Charisma adjustment.
-groundmovementadj	integer	0	Ground Movement adjustment.
-flyingadj	integer	0	Flying adjustment.
-swimmingadj	integer	0	Swimming adjustment.
-value	integer	0	Value.

### 8.1.1 class `Treasure`

[ created from class `Treasure` ]

#### **Treasure name**

[ Constructor: returns `Treasure *` ]  
Base constructor.

#### **rename obj**

[ Destructor: returns `void` ]  
Destructor.

#### **UpdateFromRecord rec**

[ Member : returns void ]  
Update Treasure from Record.

**Name**

[ Member : returns char \* ]  
Return name.

**Description**

[ Member : returns char \* ]  
Return description.

**Image**

[ Member : returns char \* ]  
Return image.

**Weight**

[ Member : returns int ]  
Return weight.

**ArmorClassAdj**

[ Member : returns int ]  
Return armor class adjustment.

**ToHitAdj**

[ Member : returns int ]  
Return to hit adjustment.

**DamageAdj**

[ Member : returns int ]  
Return damage adjustment.

**MagicalResistanceAdj**

[ Member : returns int ]  
Return magical resistance adjustment.

**DamageProtectionAdj**

[ Member : returns int ]  
Return damage protection adjustment.

**StrengthAdj**

[ Member : returns int ]  
Return strength adjustment.

**IntelligenceAdj**

[ Member : returns int ]  
Return intelligence adjustment.

**WisdomAdj**

[ Member : returns int ]  
Return wisdom adjustment.

**DexterityAdj**

[ Member : returns int ]  
Return dexterity adjustment.

**ConstitutionAdj**

[ Member : returns int ]  
Return constitution adjustment.

**CharismaAdj**

[ Member : returns int ]  
Return charisma adjustment.

**GroundMovementAdj**

[ Member : returns int ]  
Return ground movement adjustment.

**FlyingAdj**

[ Member : returns int ]  
Return flying adjustment.

**SwimmingAdj**

[ Member : returns int ]  
Return swimming adjustment.

#### Value

[ Member : returns int ]  
Return value.

#### RawData

[ Member : returns Record \* ]  
Type conversion: convert to a Record.

## 8.2 TrickTrap Class

Tricks and Traps are used to protect treasure and to generally keep the players on their toes.

#### NewTrickTrap args

[ Native method : NewTrickTrap ]

This is the interface to the constructor for TrickTrap objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the TrickTrap function with a **-this** argument to do that: [TrickTrap -this [NewTrickTrap options...]].

The defined options for this function are:

flag	type	default	explanation
-name	string	""	Name of the trick or trap.
-type	string	""	Type of the trick or trap.
-description	string	""	Description of trick or trap.
-image	string	""	Image file (GIF) of trick or trap.

### 8.2.1 class TrickTrap

[ created from class TrickTrap ]

#### TrickTrap name

[ Constructor: returns TrickTrap \* ]  
Constructor.

**UpdateFromRecord rec**

[ Member : returns void ]  
Update Treasure from Record.

**rename obj**

[ Destructor: returns void ]  
Destructor.

**Name**

[ Member : returns char \* ]  
Return name.

**Type**

[ Member : returns char \* ]  
Return trick or trap type.

**Description**

[ Member : returns char \* ]  
Return description.

**Image**

[ Member : returns char \* ]  
Return image.

**RawData**

[ Member : returns Record \* ]  
Type conversion: convert to a Record.

**8.3 Dressing Class**

Random dungeon dressings – random odds and ends. Sometimes these things have value, but real treasures use the Treasure class.

**NewDressing args**

[ Native method : NewDressing ]

This is the interface to the constructor for **Dressing** objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the **Dressing** function with a **-this** argument to do that: **[Dressing -this [NewDressing options...]]**.

The defined options for this function are:

flag	type	default	explanation
-name	string	""	Name of the dressing.
-value	integer	0	Value of the dressing.
-description	string	""	Description of dressing.
-image	string	""	Image file (GIF) of dressing.

**8.3.1 class Dressing**

[ created from class Dressing ]

**UpdateFromRecord rec**

[ Member : returns void ]

Update Dressing from Record.

**Dressing name**

[ Constructor: returns Dressing \* ]

Constructor.

**rename obj**

[ Destructor: returns void ]

Destructor.

**Name**

[ Member : returns char \* ]

Return name.

**Description**

[ Member : returns char \* ]

Return description.

**Image**

[ Member : returns char \* ]  
 Return image.

**Value**

[ Member : returns int ]  
 Return value.

**RawData**

[ Member : returns Record \* ]  
 Type conversion: convert to a Record.

## 9 Space classes

Spaces – Squares and Hexes, where things happen.

### 9.1 Geometric constants.

#### 9.1.1 class GeoConstants

[ created from class GeoConstants ]

**\$GeoConstants.Width = 100.0**

[ Constant : double ]  
 Space “width”.

**\$GeoConstants.HexSideLength = 57.735**

[ Constant : double ]  
 Side of a hex (computed to give a width of 100).

**\$GeoConstants.HexPeakHeight = 28.8675**

[ Constant : double ]  
 Height of peak above vertical sides.

## 9.2 class Exit

Exit points and other inter-spatial interconnection points like windows and staircases.

### 9.2.1 class Exit

[ created from class Exit ]

#### XCenter

[ Member : returns double ]  
Return center x coordinate.

#### YCenter

[ Member : returns double ]  
Return center y coordinate.

#### WallAligned

[ Member : returns bool ]  
Return wall alignment flag.

#### Description

[ Member : returns char \* ]  
Return description.

#### Image

[ Member : returns char \* ]  
Return picture of exit.

#### NextSpaceIndexString

[ Member : returns char \* ]  
Return next space filename.

#### Exit name

[ Constructor: returns Exit \* ]  
Constructor.

**rename obj**

[ Destructor: returns void ]  
Destructor.

**Type**

[ Member : returns char \* ]  
Return type of exit.

**9.3 ExitVector Class****9.3.1 class ExitVector**

[ created from class ExitVector ]

**ExitVector name**

[ Constructor: returns ExitVector \* ]  
Constructor.

**rename obj**

[ Destructor: returns void ]  
Destructor.

**Index index**

[ Member : returns Exit \* ]  
Index function.

**Nearest x y**

[ Member : returns Exit \* ]  
Select nearest to (x,y) function.

**ElementCount**

[ Member : returns int ]  
Return element count.

## 9.4 Item Class

### 9.4.1 class Item

[ created from class Item ]

#### XCenter

[ Member : returns double ]  
Return center x coordinate.

#### YCenter

[ Member : returns double ]  
Return center y coordinate.

#### Image

[ Member : returns char \* ]  
Return image.

#### Filename

[ Member : returns char \* ]  
Return file name.

#### Item name

[ Constructor: returns Item \* ]  
Constructor.

#### rename obj

[ Destructor: returns void ]  
Item in space.

#### Type

[ Member : returns char \* ]  
Return type of item.

## 9.5 ItemVector Class

### 9.5.1 class ItemVector

[ created from class ItemVector ]

#### **ItemVector name**

[ Constructor: returns ItemVector \* ]  
Constructor.

#### **rename obj**

[ Destructor: returns void ]  
Destructor.

#### **Index index**

[ Member : returns Item \* ]  
Index function.

#### **Nearest x y**

[ Member : returns Item \* ]  
Select nearest to (x,y) function.

#### **ElementCount**

[ Member : returns int ]  
Return element count.

## 9.6 Space Class

### NewSpace args

[ Native method : NewSpace ]

This is the interface to the constructor for **Space** objects. The argument list is an option-value list. This function does not make a first class Tcl object. You need to use the **Space** function with a **-this** argument to do that: **[Space -this [NewSpace options...]]**.

The defined options for this function are:

flag	type	default	explanation
-shape	enum SpaceShape	Space::Undefined	Space shape.
-xcenter	double	0.0	X coordinate of the center.
-ycenter	double	0.0	Y coordinate of the center.
-name	string	""	Name of space.
-description	string	""	Description of space.
-backgroundcolor	string	"white"	Background color of space.

### 9.6.1 class Space

[ created from class Space ]

#### UpdateFromRecord rec

[ Member : returns void ]

Update Space from Record.

#### Space name

[ Constructor: returns Space \* ]

Constructor.

#### rename obj

[ Destructor: returns void ]

Destructor.

#### CenterX

[ Member : returns double ]

Return this space's center point X coordinate.

**SetCenterX newX**

[ Member : returns double ]

Set the space's center point X coordinate.

**CenterY**

[ Member : returns double ]

Return this space's center point Y coordinate.

**SetCenterY newY**

[ Member : returns double ]

Set the space's center point Y coordinate.

**NearestExit x y**

[ Member : returns Exit \* ]

Return the nearest exit.

**IndexedExit index**

[ Member : returns Exit \* ]

Return the ith exit.

**NumberOfExits**

[ Member : returns int ]

Return the count of exits.

**InsertExit source**

[ Member : returns void ]

Insert an exit.

**DeleteExitNear x y**

[ Member : returns void ]

Delete exit nearest to x,y.

**DeleteExitAtIndex index**

[ Member : returns void ]

Delete exit by index.

**NearestItem x y**

[ Member : returns Item \* ]

Return the nearest item.

**IndexedItem index**

[ Member : returns Item \* ]

Return the ith item.

**NumberOfItems**

[ Member : returns int ]

Return the count of items.

**InsertItem source**

[ Member : returns void ]

Insert an item.

**DeleteItemNear x y**

[ Member : returns void ]

Delete item nearest to x,y.

**DeleteItemAtIndex index**

[ Member : returns void ]

Delete item by index.

**Name**

[ Member : returns char \* ]

Return name of the space.

**SetName newN**

[ Member : returns char \* ]

Set the name of the space.

**Description**

[ Member : returns char \* ]

Return description of the space.

**SetDescription newD**

[ Member : returns char \* ]

Set the description of the space.

**BackgroundColor**

[ Member : returns char \* ]

Return background color of the space.

**SetBackgroundColor newBG**

[ Member : returns char \* ]

Basic Space class.

**Shape**

[ Member : returns char \* ]

Return this space's shape.

**SetShape newS**

[ Member : returns int ]

Set this space's shape.

**RawData**

[ Member : returns Record \* ]

Type conversion: convert to a Record.

**InsertNewExit ?type? ?x? ?y? ?wa? ?d? ?im? ?ns?**

[ Member : returns int ]

Insert a new exit.

**InsertNewItem ?type? ?x? ?y? ?im? ?fn?**

[ Member : returns int ]

Insert a new item.

**MakeGraphicCommmand ?scaleFactor?**

[ Member : returns int ]

Generate the command body to draw the base space:  
eval [concat .canvas create [space MakeGraphicCommmand ?scale?]].

## References

- [1] Gary Gygax. *Monster Manual*. TSR Games, Lake Geneva, WI, 1977, 1978.
- [2] Gary Gygax. *Players Handbook*. TSR Games, Lake Geneva, WI, 1978.
- [3] Gary Gygax. *Dungeon Masters Guide*. TSR Games, Lake Geneva, WI, 1979.

## Index

Character Class, 4

Copyright Info, 2

Dice Class, 3

Dressing Class, 25

Exit Class, 28

Item Class, 30

Monster Class, 11

Record Class, 3

Space Class, 27

Spell Class, 17

Treasure Class, 20

TrickTrap Class, 24