Aas advanced authoring system

Table of Contents

Introduction

In the 1980s, a type of computer game known colloquially as "text adventure" were among the most popular. Players quickly learned to type such commands as **N** and **LOOK DRAGON**. But with time, companies stopped selling text adventures. Text adventures languished, a forgotten tidepool in the giant rushing river that is the ever-changing computer gaming industry.

Languished, that is, until now! The Advanced Authoring System allows you to create text adventures (or "TAs," to those in the know) quickly and easily. No programming is required, and thanks to its use of the eXtensible Markup Language, XML, Advanced Authoring System games can be played on any number of platforms. XML insures that your game will always be readable. Furthermore, Advanced Authoring System is written in Java, meaning that your TAs can be played not only on Windows, but also on Macs and even on such minority platforms as OS/2 and Linux.

But these technical details may not interest you. "Roddy," you may be saying, "what does all of this have to do with writing text adventures?" Right, sure. What makes it easy to write TAs is AASide, a user-friendly graphical user interface that makes writing text adventures a snap. All you have to do is fill in text fields, click check boxes, and choose items from a list. You might think that all of this user-friendliness comes at the expense of power. Not at all. Through AAS's system of countdowns, you can create a flexible text adventure to rival the text adventures of old, and all without programming.

If you want to make sure you have the latest version of AAS, be sure to visit the website at http://www.aas-ta.com/.

And before I go too much further in to the manual, I need to give thanks where it's deserved. Linc, CandyFox, and ninjaschlong, from the AAS forum, provided much-needed proofreading, since I'm not the best writer in the world. And many thanks go to David Banner, who read through the final draft and even contributed a whole chapter!

Now hold on to your hat and get ready to experience the power of AAS!

Roddy Ramieson

Chapter I: Starting Your First Game

Chances are you're itching to get started. Well, go ahead! AAS is so easy to use, you should be able to create your first game without having to read the entire manual. By following this short tutorial, soon you'll be adventuring away.

Be sure to notice that the game I develop in this chapter, *Cave of Adventure*, is included with the documentation. That makes it easy to follow along with me.

Starting AASide

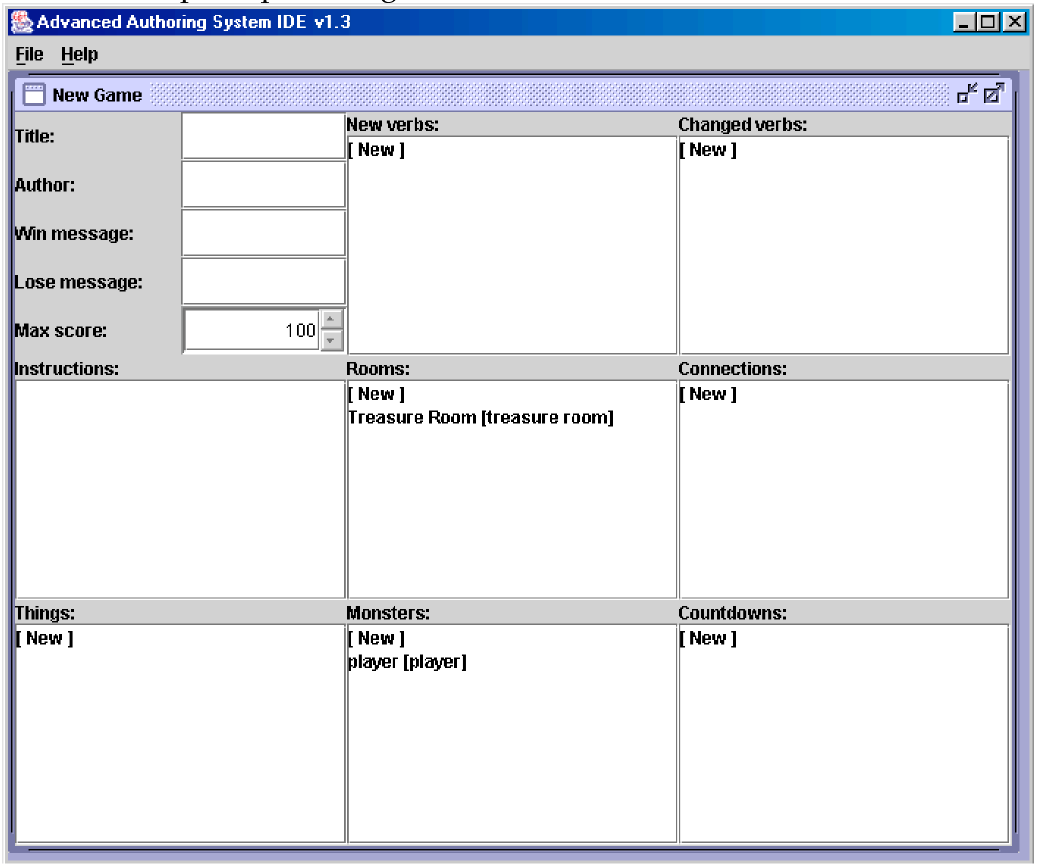
If you've downloaded one of the full installation packages, you should be able to double-click the AASide icon to start it. If you downloaded the Java version, first make sure you have a recent version of the Java VM, available from http://java.sun.com/. Then open a terminal window. Once you've done that, get to the folder where you put AASide. Now you're ready to run it by typing

```
    java -jar aasidexx.jar
```

xx is the version number of AASide that you downloaded.

Initial Properties

When you start AASide, it will be completely blank. Under the File menu, choose New. That will open up a new game window.



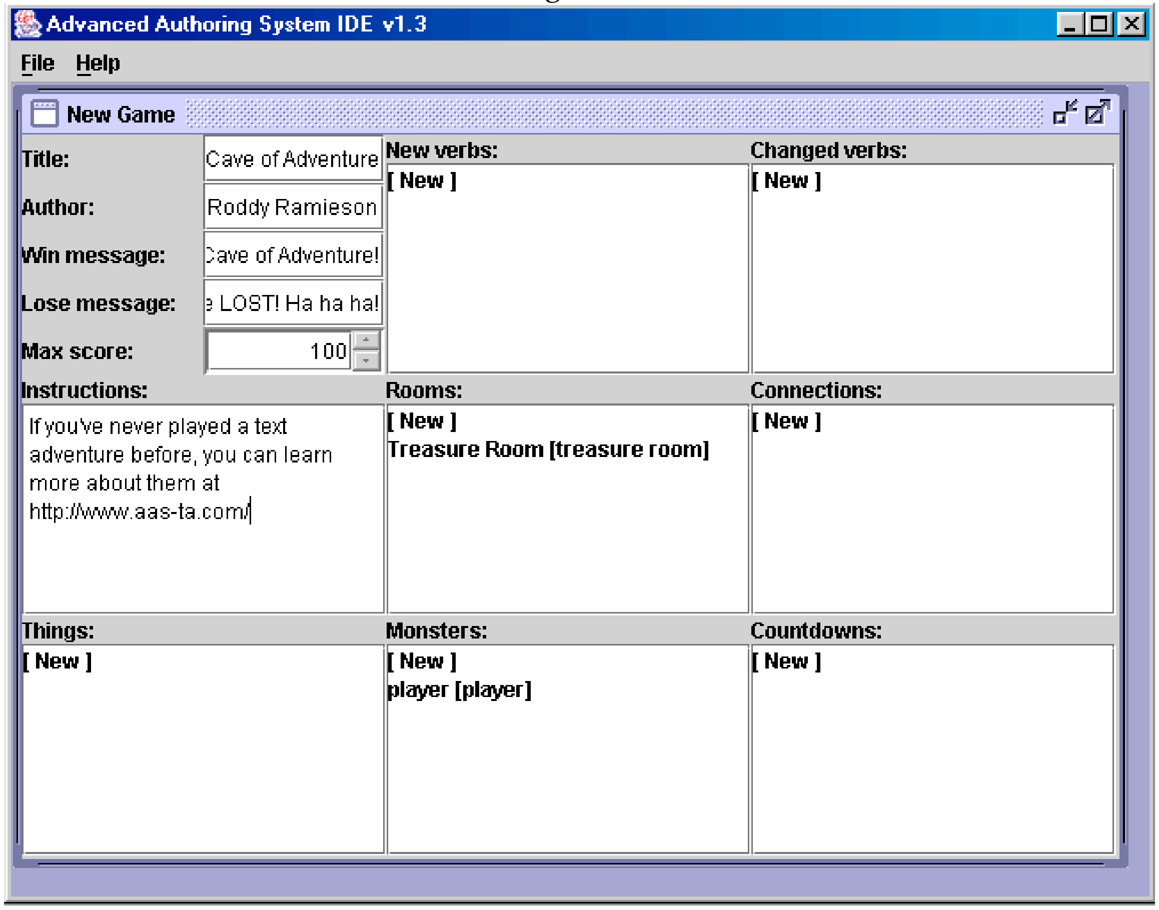
Title:
Author:
Win message:
Lose message:
Max score: 100
Instructions:
New verbs: [New]
Changed verbs: [New]
Rooms: [New] Treasure Room [treasure room]
Connections: [New]
Things: [New]
Monsters: [New] player [player]
Countdowns: [New]

A lot of this you'll undoubtedly understand. **Title** is where you put the title of your game. **Author** is where your name goes. **Win message** is what is printed when the player wins by putting all of the treasures in the treasure room. **Lose message** is what is printed when the player loses.

Let's start an example game. I'll call it *Cave of Adventure*. Type "Cave of Adventure" into the **Title** box. In the **Author** box, type your name. (I'll type mine in, of course.) Let's give the game a **Win message** like, "Congratulations! You have won Cave of Adventure!" For a **Lose message**, let's use, "You have LOST! Ha ha ha!" (You can use different messages if you like – it is your game, after all.)

Instructions is where you would normally put instructions for your game. Put in something simple. I'm going to tell people to visit http://www.aas-ta.com/ for more instructions if they've never played an adventure game before.

Your adventure should look something like this now:



Title: Cave of Adventure
Author: Roddy Ramieson
Win message: Cave of Adventure!
Lose message: LOST! Ha ha ha!
Max score: 100

Instructions:
If you've never played a text adventure before, you can learn more about them at http://www.aas-ta.com/

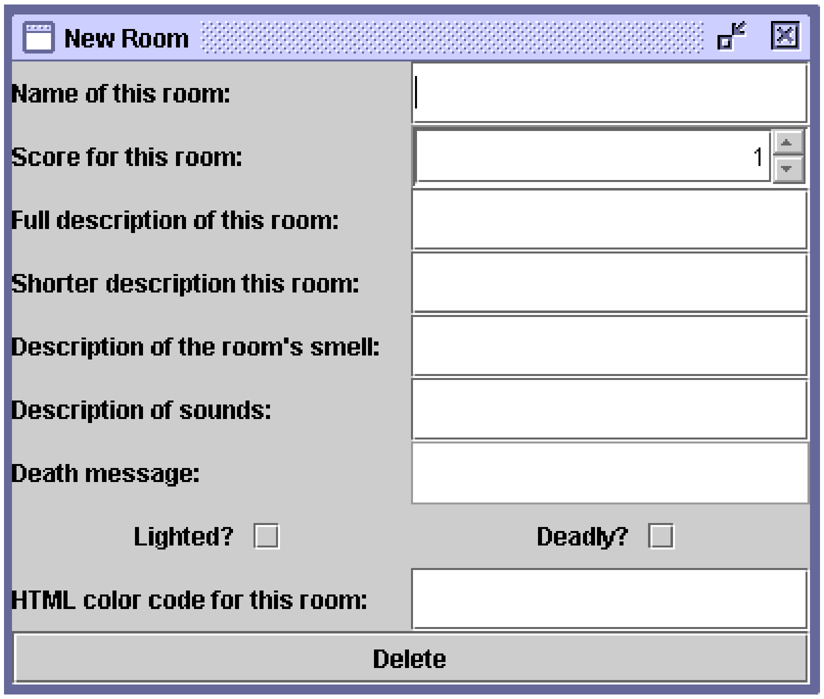
Rooms:
Treasure Room [treasure room]

Monsters:
player [player]

Before we go any further, let's save this. We wouldn't want a power glitch to wipe out all of our work! Select **File > Save As**, and save the game as `cavead.aas`.

A Few Good Rooms

We've got the start of a game, but it's not very exciting. First we need some rooms, the locations where our TA takes place. To make a new room, click on **[New]** in the **Rooms:** section. You'll get a New Room dialog.


New Room
Name of this room:
Score for this room:
1
Full description of this room:
Shorter description this room:
Description of the room's smell:
Description of sounds:
Death message:
Lighted?
Deadly?
HTML color code for this room:
Delete

The important things to notice right now are the **Name of this room**, which gives the room name, **Score for this room**, which tells how many points the player gets for making it to this room, **Full description of this room**, which is the what the player sees when (s)he looks around, **Shorter description of this room**, which is what the player sees if they don't want the full description, and **Lighted**, which controls whether or not there's light in the room. When you're done editing a room, you close the window by clicking on the X button in the upper-right-hand corner.

Let's make four rooms: Cave Entrance, Deep Inside Cave, Waterfall Room, and Crawling Room. Describe them however you like; just make sure to create all four, and have all four of them lighted. You'll also need to make the Treasure Room Lighted, as well as give it descriptions. And once you're done, don't forget to save!

Getting From Here to There

These rooms are great and all, but they're not connected in any way. That's where connections come in.

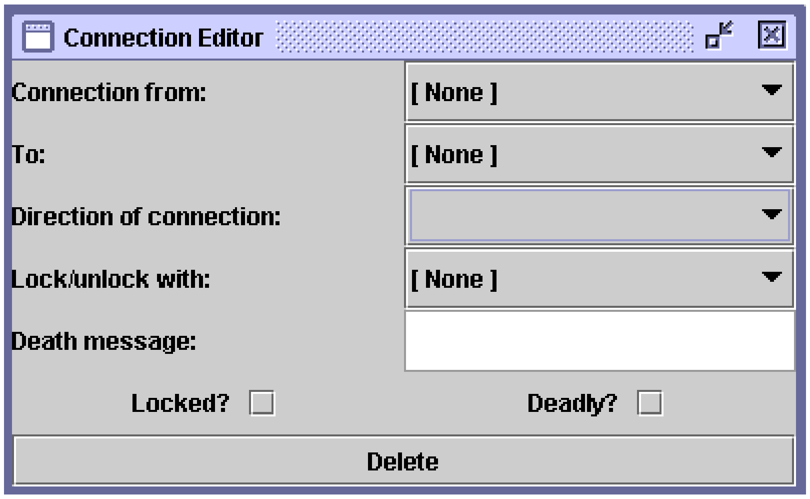
Connections are the compass directions, like north and west, that hook rooms together. For our game, we'll have the following connections:

- Cave Entrance
    - North to Deep Inside Cave

- Deep Inside Cave
  - South to Cave Entrance
  - East to Waterfall Room
  - West to Crawling Room
- Waterfall Room
  - West to Deep Inside Cave
  - North to Treasure Room
- Crawling Room
  - East to Deep Inside Cave
- Treasure Room
  - South to Waterfall Room

Notice that this list demonstrates Newton's Third Law of Text Adventure Passages: for every connection, there is (usually) an equal and opposite connection. Otherwise there wouldn't be any way to get back from a room once you arrive there.

To make a connection, click on **[New]** in the **Connections:** section. You'll get a dialog that looks like this:



Connection Editor

Connection from:	[None]
To:	[None]
Direction of connection:	
Lock/unlock with:	[None]
Death message:	

Locked? ☐ Deadly? ☐

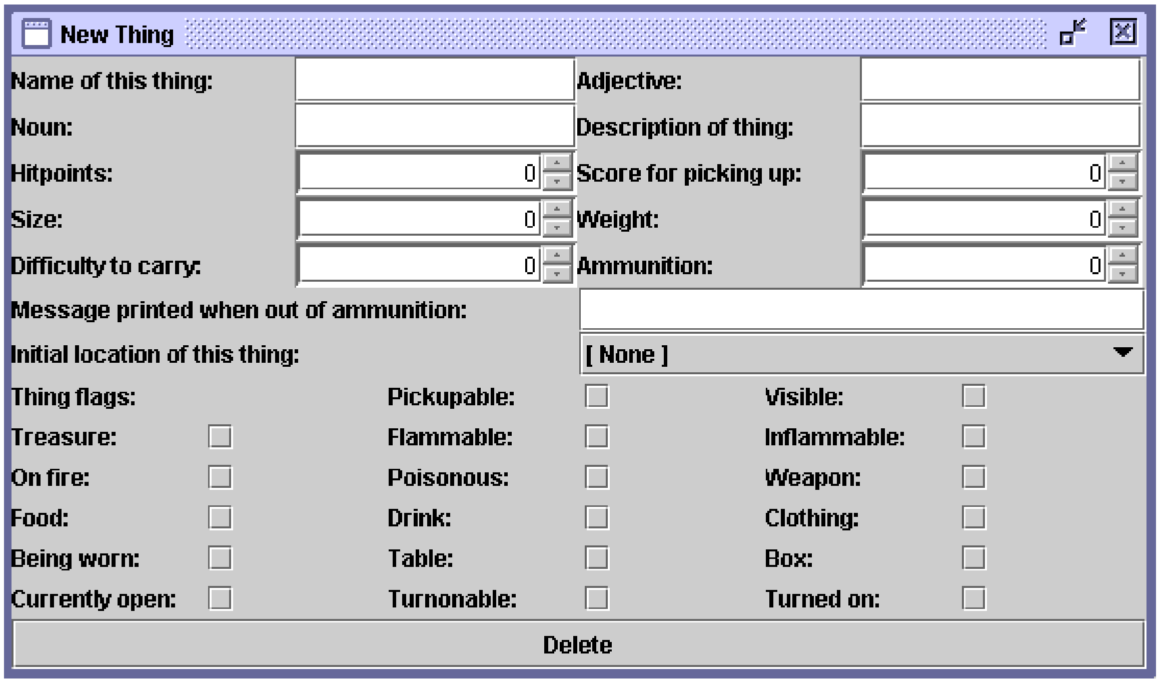
For each connection, you choose the room where the connection leads from, the room the connection leads to, and what direction the connection is in. (You can see that there's more you can do with Connections, but we won't be doing that in *Cave of Adventure*.)

Enter the eight connections listed above and you'll be good to go.

Adding Things

This cave is all well and good, but you really need something to pick up. That's half the fun of text adventures: taking everything that's not glued to the table! If nothing else, you need to have a treasure, since the game won't end until all of the treasures are in the treasure room.

So let's make a sparkly diamond. Click on **[New]** in the **Things:** section. That will bring up the Things dialog:



Name of this thing:		Adjective:	
Noun:		Description of thing:	
Hitpoints:	0	Score for picking up:	0
Size:	0	Weight:	0
Difficulty to carry:	0	Ammunition:	0
Message printed when out of ammunition:			
Initial location of this thing:	[None]		

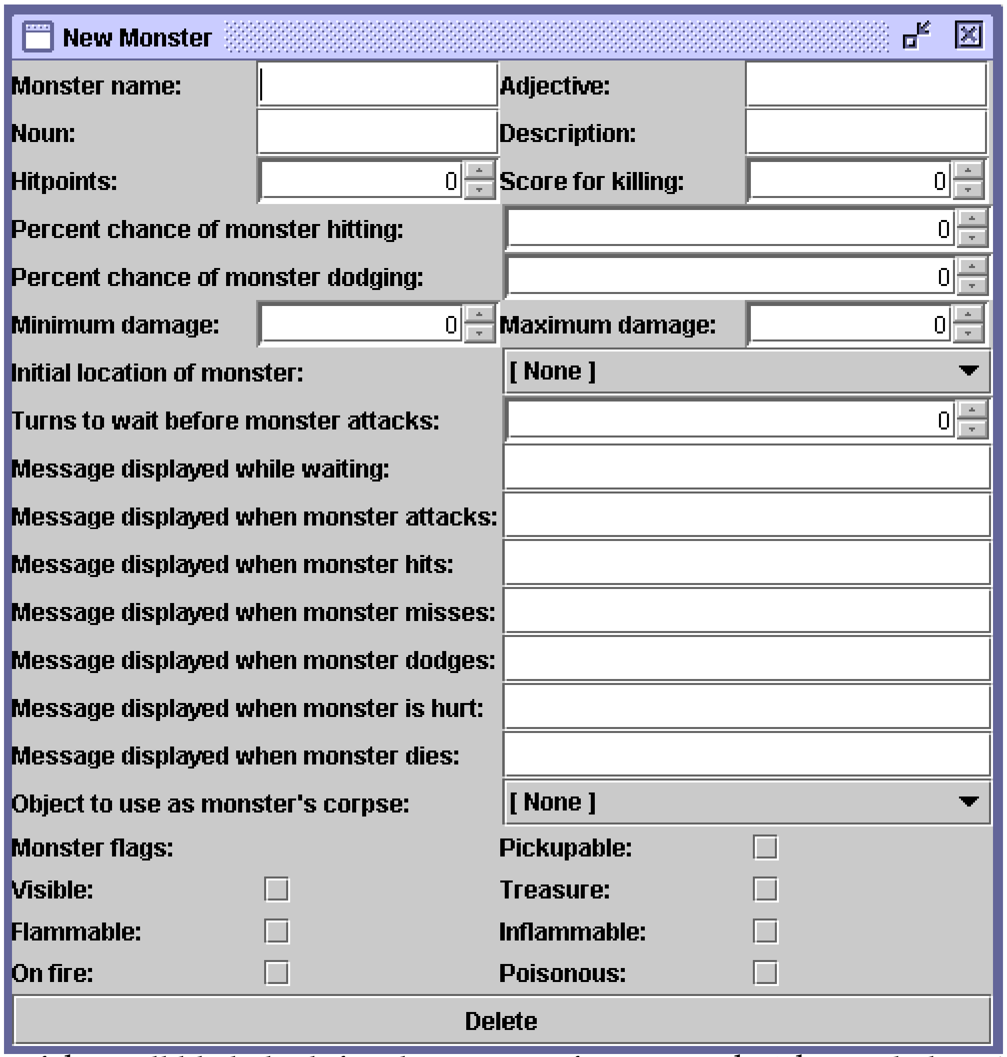
Thing flags:		Pickupable:	☐	Visible:	☐
Treasure:	☐	Flammable:	☐	Inflammable:	☐
On fire:	☐	Poisonous:	☐	Weapon:	☐
Food:	☐	Drink:	☐	Clothing:	☐
Being worn:	☐	Table:	☐	Box:	☐
Currently open:	☐	Turnonable:	☐	Turned on:	☐

You can do a lot in this dialog, far more than I can cover in this brief introduction. Instead, let me tell you exactly what you need to do to make a diamond. In **Name of this thing** put "Sparkly Diamond." In **Adjective** put "sparkly". In **Noun** put "diamond". For its **Initial location**, select Crawling Room. Click **Treasure** and **Pickupable**, to indicate that it's a treasure that can be picked up, and **Visible**, so the player can see the diamond. And that's it!

Live Challenges

We've added an inanimate object to our game; now let's add an animated object. TAs usually have monsters, though "monster" is really a generic term. Notice that even the player is a monster.

But this is a cave, so the player won't be surprised to find an angry bat! To create the bat, click on **[New]** in the **Monsters:** section. That will bring up the Monsters dialog:



New Monster

Monster name:		Adjective:	
Noun:		Description:	
Hitpoints:	0	Score for killing:	0
Percent chance of monster hitting:	0		
Percent chance of monster dodging:	0		
Minimum damage:	0	Maximum damage:	0
Initial location of monster:	[None]		
Turns to wait before monster attacks:	0		
Message displayed while waiting:			
Message displayed when monster attacks:			
Message displayed when monster hits:			
Message displayed when monster misses:			
Message displayed when monster dodges:			
Message displayed when monster is hurt:			
Message displayed when monster dies:			
Object to use as monster's corpse:	[None]		
Monster flags:		Pickupable:	☐
Visible:	☐	Treasure:	☐
Flammable:	☐	Inflammable:	☐
On fire:	☐	Poisonous:	☐

Delete

Much of this will likely look familiar to you after seeing the Things dialog. A few things bear explaining. Monsters have a percent chance to hit the player and a percent chance to dodge the player's blows. The damage the monster does will be a random amount between the **Minimum damage** and **Maximum damage** amounts.

When the player enters the same room as the monster, it will attack after a number of turns equal to **Turns to wait before monster attacks**. The messages are self-explanatory, except for one thing: you have to add a space to the end of each message. You can have an object that appears as the monster's corpse after it's killed.

Let's make an evil bat that will protect the Waterfall Room. Make its name "evil bat", with an adjective of "evil" and a noun of "bat". Make its hitpoints 5. Let it wait 2 turns before attacking. Give it a minimum damage of 2 and a maximum damage of 4 – there's only so much damage a bat can do. Let it have a hit percentage of 30% and a dodge of 40%, since bats can always fly away from a player's clumsy swings. Add in some messages (add an extra space to the end of each message), and make sure it's visible.

While we're adjusting monsters, let's set the player's attributes. Make him/her start in the Cave Entrance. Let the player have 10 hit points, with a percent chance of hitting equal to 80%, and a dodge of 50%. Set a minimum damage of 2 and a maximum of 8. You can also set the player's description, which the player will see if (s)he types **EXAMINE ME**.

Playing the Game

We now have a completable game, albeit a simple one. To play it, open a terminal window. Once you've done that, get to the folder where you put AAS (and, hopefully, `cavead.aas`). Now you're ready to play the game by typing

```
     java –jar AASexe.jar cavead.aas
```

Congratulations! You've now created your first AAS game!

Chapter II: Basic Elements of AAS

An AAS game consists of a number of different elements.

Rooms. Rooms are the foundation of the building that is your AAS game. They are the locations where your game takes place. They are the bedroom in your house, the morgue in the hospital, Pressure Dome #3 in the Mars colony.

Connections. If rooms form the foundation, connections are the mortar that holds the foundation blocks together. Connections are how you get from one room to another.

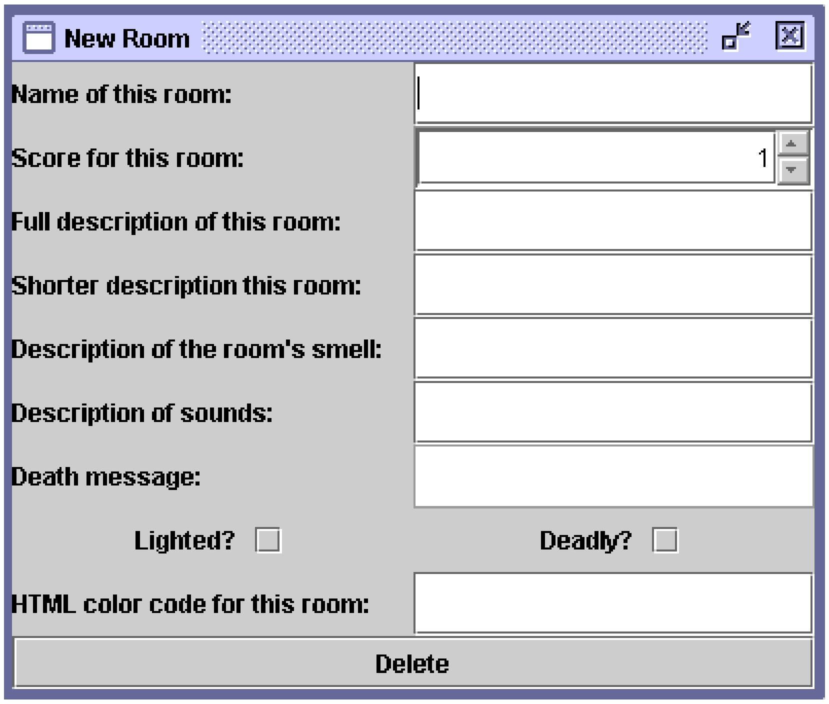
Things. If rooms are the foundation and connections the mortar, then things are the walls of your AAS game. A game without things is like a gazebo: fun to visit when you're in the mood for a snog, but you wouldn't want to live there. Tables, books, bottles, all are things.

Monsters. If rooms are the foundation and connections the mortar and things the walls, then monsters are the parts of the floor that are uneven and that you trip over. Monsters are the living things in your game, whether the friendly mailman or the evil nameless horror from the depths that wants to rip out your spleen.

Countdowns. While AAS is designed to handle nearly everything you need automatically, there are times when you will need something different than the default behavior. That's where countdowns come in. Using them, you can have events occur at a scheduled time, or when the player picks up an object. You can even string multiple countdowns together for unlimited complexity. Countdowns will be covered in the next chapter.

Rooms

You create rooms in AASide by clicking on **[New]** in the **Rooms** section. When you do, the New Room dialog will appear.



Name of this room:	
Score for this room:	1
Full description of this room:	
Shorter description this room:	
Description of the room's smell:	
Description of sounds:	
Death message:	
Lighted? ☐	Deadly? ☐
HTML color code for this room:	

There is one special room in AAS games, the Treasure Room. The Treasure Room is where the player is to drop all of the treasures (s)he collects. The game isn't over until all treasures are in the treasure room. You can call the room whatever you want; AASide will mark the Treasure Room with **[treasure room]** so you can keep up with it.

Room Descriptions

The most important element of a room is its **Name**. The name is printed every time a player enters the room, and every time the player looks around. Even if the player has chosen not to see the other room descriptions, (s)he will see the room name.

AAS games allow you to describe a room using the senses of sight, smell, and hearing. For sight, there are the **Full description** and **Shorter description**. The full description is the main description of the room, the main way the player learns about the room. If the player is playing in **VERBOSE** mode, the player will see the full description every time (s)he enters the room. The shorter description is shown if the player is in **SEMIVERBOSE** mode, and should be a more compressed version of the full description.

For smell, there is the **Description of the room's smell**. It is shown when the player types **SMELL**. For hearing, there is the **Description of sounds**, shown when the player types **LISTEN**.

Other Room Elements

The **Score for this room** is the number of points the player gets for visiting the room. You can use this as a reward for a player reaching a particularly hard-to-reach or guarded room.

If the room is not **Lighted**, then it is dark, and the player won't be able to see anything without a light source. If the room is **Deadly**, then the player is shown the **Death message** and then killed.

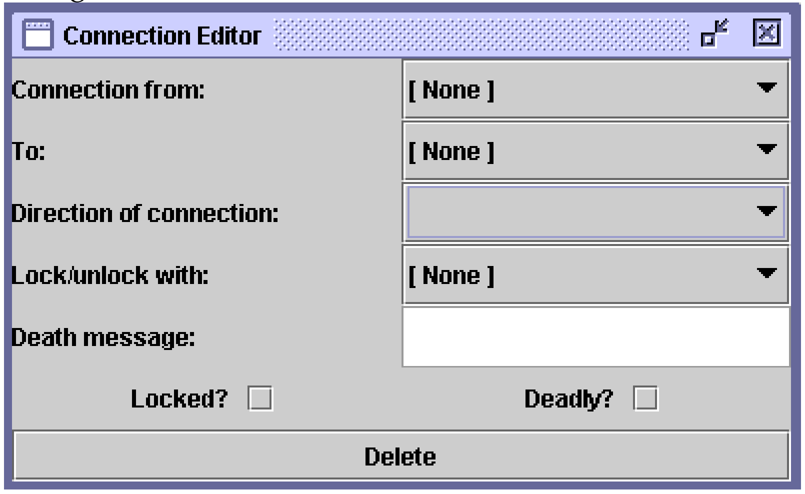
Finally, if you'd like, you can set the background color of the room using HTML codes, which are RRGGBB, where R is the red value, G is the green value, and B is the blue value, in hexadecimal.

Connections

Connections can lead in the cardinal directions north, south, east, and west, as well as up and down. They go from room to room, and are one-way. In most case you'll used paired connections. For example, one connection will go north from room 1 to 2; the second will go south from room 2 to 1.

You might ask, why not have connections be automatically two ways? This way, you can code one-way connections, like a laundry chute that dumps the player into a basement, or have connections that are different in opposite directions, like directions in a maze.

Clicking on **[New]** in the **Connections** section will bring up the New Connection dialog window.


Connection Editor

Connection from:	[None]
To:	[None]
Direction of connection:	
Lock/unlock with:	[None]
Death message:	
Locked? ☐	Deadly? ☐

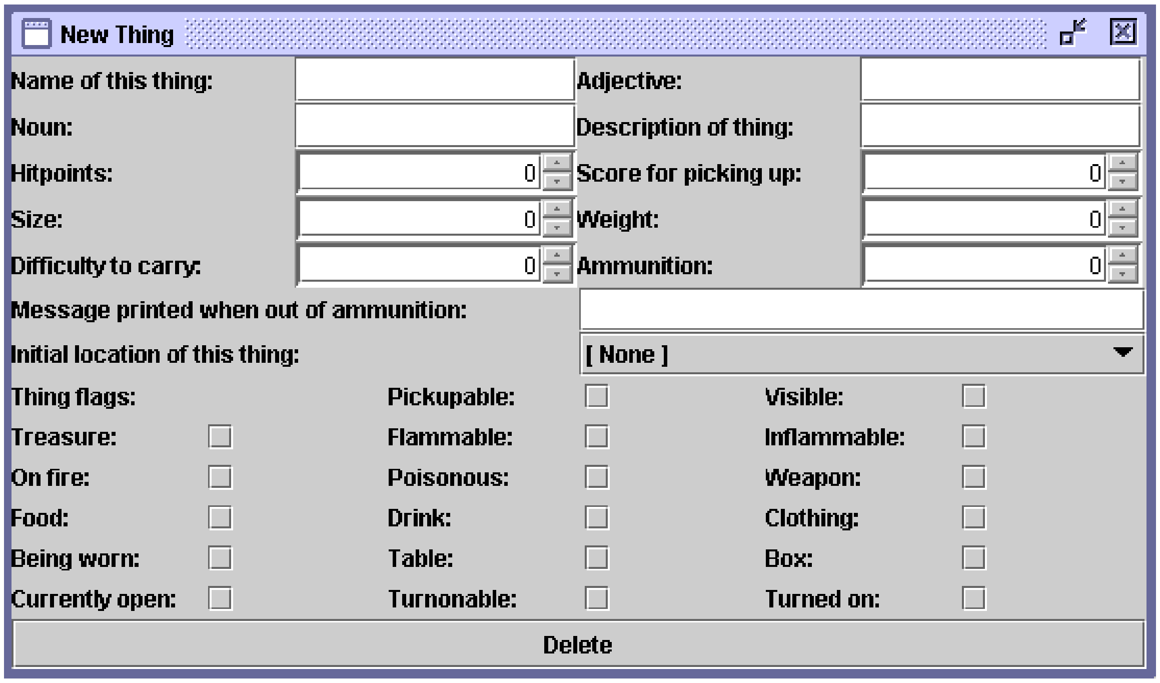
A connection goes **from** a given room **to** a second room. The **Direction of connection** is the cardinal direction (or up or down) in which the direction goes.

Connections may be **Deadly**. If so, going in that direction kills the player immediately, and prints the **Death message**.

Connections can also be locked, if they are doors or something like that. The **Lock/unlock with** list lets you choose a thing that will be the key that locks or unlocks the connection. If **Locked** is checked, then the connection begins locked.

Things

Things do a lot of things, causing them to be not quite as easy to create as rooms or connections. Click on **[New]** in the Things section to bring up the New Thing dialog.



New Thing

Name of this thing:		Adjective:	
Noun:		Description of thing:	
Hitpoints:	0	Score for picking up:	0
Size:	0	Weight:	0
Difficulty to carry:	0	Ammunition:	0
Message printed when out of ammunition:			
Initial location of this thing:	[None]		

Thing flags:		Pickupable:	☐	Visible:	☐
Treasure:	☐	Flammable:	☐	Inflammable:	☐
On fire:	☐	Poisonous:	☐	Weapon:	☐
Food:	☐	Drink:	☐	Clothing:	☐
Being worn:	☐	Table:	☐	Box:	☐
Currently open:	☐	Turnonable:	☐	Turned on:	☐

Basic Details

The **Name of this thing** is what's given in the room description and when the player picks up the thing (assuming it can be picked up). Things can have an **Adjective**, such as "red," and a **Noun**, such as "marble." The **Description of thing** is what's printed when the player looks at the thing.

There are three things determining how easily the player can carry the thing: its **Size**, **Weight**, and its **Difficulty to carry**. The values are the percentage of the total amount the player can handle. For example, a large beach ball might have a weight of 5 (meaning 5% of the total weight the player can carry), a size of 80 (because it's big), and a carry difficulty of 50 (because it's slick and doesn't have any handles).

The **Score for picking up** is how much the player gets for having the thing in his/her hands. If (s)he drops the thing, the points go away again. The thing's **Initial location** is where it starts the game out. This can be a room, another thing (like a table), a monster, or even the player.

Hitpoints

A thing's hitpoints determine how hardy the thing is. If a thing is on fire, it loses several hitpoints every turn. If the player hits a thing, it loses hitpoints. Once a thing's hitpoints drop to 0, the thing is destroyed.

Hitpoints also play a role in edible and drinkable things. When the player eats or drinks a thing, it transfers its hitpoints to the player, healing him/her.

Flags

Things can have a number of flags:

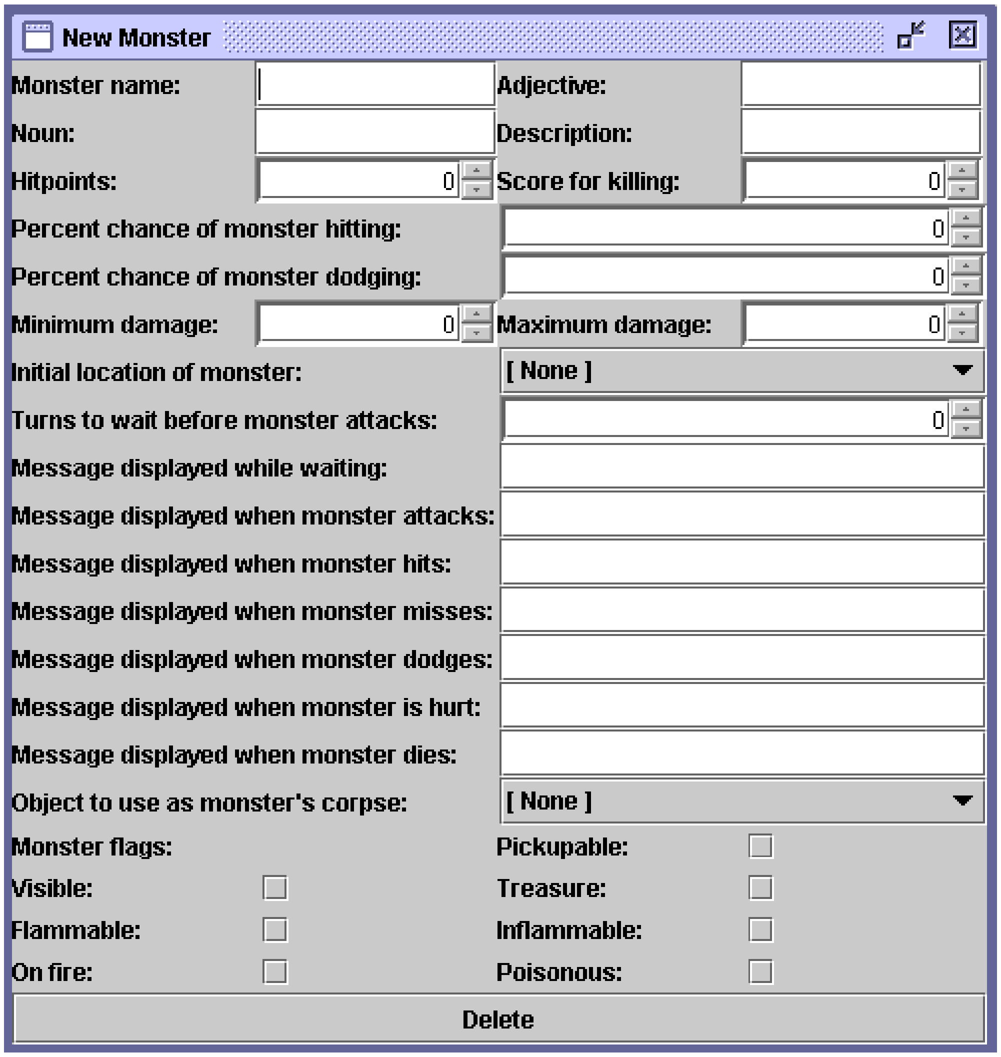
- **Pickupable**. The thing can be picked up by the player.
- **Visible**. The thing can be seen by the player.
- **Treasure**. The thing is a treasure object. The game doesn't end until all treasures are in the Treasure Room.
- **Flammable**. The thing can catch fire, losing 10 hitpoints a turn.
- **Inflammable**. The thing catches fire so easily, it loses 100 hitpoints a turn.
- **On fire**. The thing starts the game on fire.
- **Poisonous**. The thing is poison. If the player eats or drinks it, (s)he will lose 3 hitpoints a turn until dead, unless overridden by a countdown.
- **Weapon**. The thing can be used as a weapon.
- **Food**. The thing can be eaten.
- **Drink**. The thing can be drunk.
- **Clothing**. The thing can be worn.
- **Being worn**. At the start of the game, the thing is being worn.
- **Table**. You can put other things on the thing.
- **Box**. You can open and close the thing.
- **Currently open**. The thing starts the game open.
- **Turnonable**. The thing can be turned on.
- **Turned on**. The thing starts the game turned on.

Weapon Things

When things are weapons, they have **Ammunition**, which tells how much you can use the weapon. After the ammunition reaches zero, the weapon can't be used any more. When the weapon runs out of ammo, the **Message printed when out of ammunition** is printed. If you want a weapon that doesn't use ammunition, you can set the ammunition to 9999.

Monsters

To create a monster, click on **[New]** in the Monsters section to bring up the New Monster dialog.


New Monster

Monster name:		Adjective:	
Noun:		Description:	
Hitpoints:	0	Score for killing:	0
Percent chance of monster hitting:	0		
Percent chance of monster dodging:	0		
Minimum damage:	0	Maximum damage:	0
Initial location of monster:	[None]		
Turns to wait before monster attacks:	0		
Message displayed while waiting:			
Message displayed when monster attacks:			
Message displayed when monster hits:			
Message displayed when monster misses:			
Message displayed when monster dodges:			
Message displayed when monster is hurt:			
Message displayed when monster dies:			
Object to use as monster's corpse:	[None]		
Monster flags:		Pickupable:	☐
Visible:	☐	Treasure:	☐
Flammable:	☐	Inflammable:	☐
On fire:	☐	Poisonous:	☐

There is one special monster in every AAS game, the player. The player can have all of the same properties of any other monster.

Basic Details

The **Monster name** is what's given in the room description. Monsters, like things, can have an **Adjective**, such as "wooden," and a **Noun**, such as "boy." The **Description** is what's printed when the player looks at the monster. The **Initial location of monster** is what room the monster is in at the start of the game.

Flags

Monsters can have a number of flags:

- **Pickupable**. The monster can be picked up. This is most useful if the monster is a puppy, or perhaps a pocket dragon.
- **Visible**. The monster can be seen.
- **Treasure**. The monster is a treasure, and must be in the Treasure Room for the game to end.

- **Flammable**. The monster can catch fire. For every turn the monster is on fire, it will lose 10 hitpoints.
- **Inflammable**. The monster can *really* catch fire. Every turn it's on fire, the monster loses 100 hitpoints.
- **On fire**. At the start of the game, the monster is on fire.
- **Poisonous**. The monster is poisonous. If the monster hits the player, the player will be poisoned.

Combat

Several of the fields in the New Monster dialog window are for combat, so now a word on it. Combat is as simple as can be: the attacker has a percent chance to hit. If the attacker hits, then the defender gets a percent chance to dodge. If the defender does not dodge, then the defender takes a random amount of damage between the monster's minimum and maximum damage.

Given that, you undoubtedly understand more about the fields in the New Monster dialog window. The monster's **Hitpoints** are a measure of its health. The **Score for killing** is how many points the player earns for killing the monster. The **Percent chance of monster hitting** and **dodging** are the probability that it will hit an opponent or dodge an opponent's blow. Its **Minimum** and **Maximum damage** are the limits to the amount of damage it will deal in any one blow.

When a player enters a room, the monster will wait before attacking the player. How long it waits is controlled by the **Turns to wait before monster attacks** field. If you want a monster to be friendly, just set this number to a high value, such as 9999.

There are a number of messages that are printed during combat. The only other thing you need to know is that every message should end with an extra space, so that the messages look right when strung together.

When the monster dies, it can optionally leave a corpse, so the player can do that looting that players are so fond of. Set it using the **Object to use as monster's corpse** list.

Chapter III: Countdowns, Triggers, Events, Conditions

While AAS handles most of the standard things you'll undoubtedly want to do, on occasion you may need to do something more complicated. That's what countdowns are for.

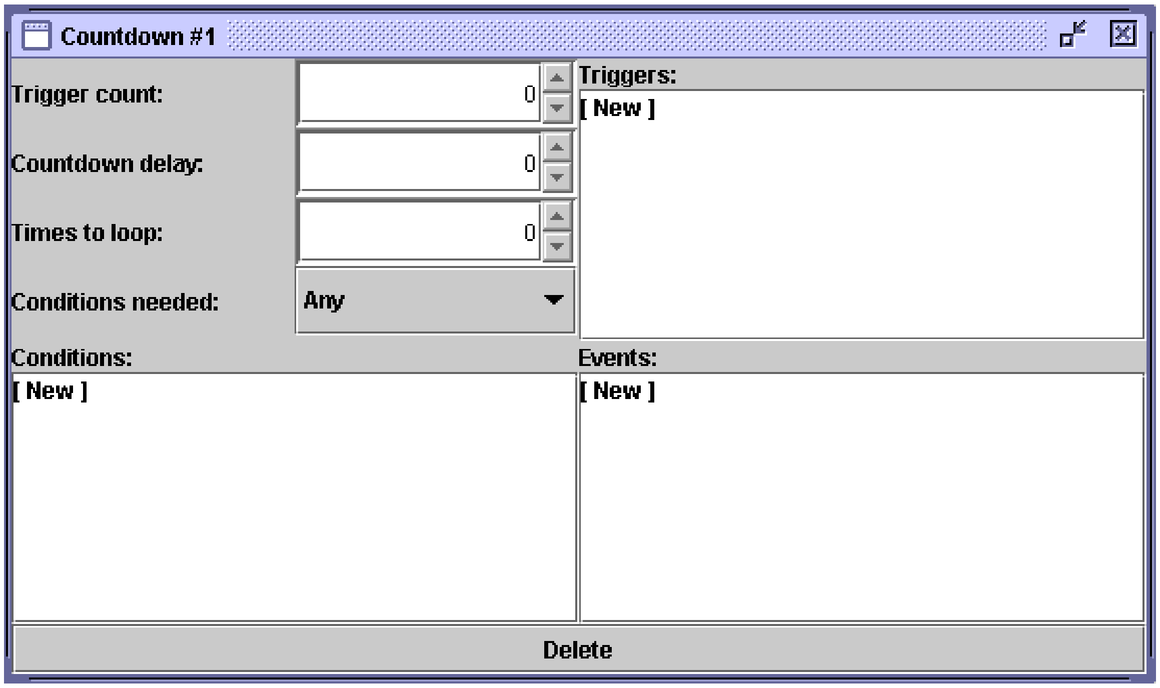
In the misty mists of time, during the early days of AAS, countdowns were used to trigger events after a certain number of turns. These days countdowns can do much more, through the combination of conditions, triggers, and events.

Conditions are what must be true for the countdown to occur. They handle things such as, "this thing must be in this other thing," or, "the player's score must be greater than a certain number." Conditions can be inverted, allowing a full range of conditional logic. (If you're unfamiliar with "conditional logic," I'd suggest a quick trip to Google.)

Triggers are what causes the countdown to happen. Triggers include things such as "the player has entered a certain room," or, "the monster's hit points have dropped below a certain level."

Events are what happens when the countdown occurs. Events are things like, "move a thing to a new room," or, "change a connection's destination to another room," or even, "trigger another countdown."

Clicking **[New]** in the **Countdown** menu will bring up the countdown dialog window.



Trigger count: 0
Countdown delay: 0
Times to loop: 0
Conditions needed: Any

The **Trigger count** is how many times the triggers must occur before the countdown's events take place. The **Countdown delay** is how many turns to wait after the trigger occurs before having the countdown's events take place. **Times to loop** is how many times the events should take place. A loop time of 0 is the same as a loop time of 1, and means the events should take place only once.

Conditions

Conditions are what must be true for a countdown to occur. If a countdown's conditions aren't true, then it won't happen, regardless of anything else.

Countdowns don't have to have conditions. If the countdown doesn't have conditions, then it will occur as soon as its triggers are met.

A countdown can have any number of conditions. The countdown can occur when any of the conditions are true, or only when all of the conditions are true. (These are what us snooty computer science types call OR and AND logic.) To control whether all of the conditions must be true or whether any of the conditions can be true, set the **Conditions needed** field.

Each individual condition can be inverted, reversing its normal meaning.

AAS supports the following conditions.

- **isopen**. True if a thing is open.
- **isdead**. True if a monster is dead.
- **isin**. True if one thing is in another thing, monster, or room.
- **isvisible**. True if a monster or thing is visible from a given object, monster, or room. The latter will often be the player, letting you check to see if the player can see a given thing.
- **iseaten**. True if a given thing has been eaten.
- **ison**. True if a given thing is currently turned on.
- **isworn**. True if a given thing is currently being worn.
- **isscoreabove**. True if the player's score is above a certain value.
- **isprob**. True if a random probability is equal to or less than the given probability.
- **isonfire**. True if a given thing or monster is on fire.
- **ispoisoned**. True if a given monster is currently poisoned.

Triggers

Triggers cause the countdown to occur. The triggers correspond to changes in the game. Strictly speaking, you don't need a trigger. If you don't have any, then the only way for the countdown to occur is for it to be caused by another countdown's events.

All triggers can be true either the first time the trigger occurs, or every time the trigger occurs. If the trigger's **Trigger first time only** value is set, then the trigger will only fire the first time. I know that's a little complex, so let me give you an example. Suppose you're going to have a trigger that fires when a light is turned on. If you set the **Trigger first time only** value, then the countdown will be triggered the first time the lamp is turned on. If you don't set it, then the countdown will be triggered every time the lamp is turned on.

AAS supports the following triggers. With each trigger, the countdown will be triggered when:

- **onattack**. A given monster is attacked.
- **ondouse**. A given thing that is lit is doused.
- **onturn**. The game reaches a specific turn.
- **onturnoff**. A given thing is turned off.
- **onturnon**. A given thing is turned on.
- **onlock**. A given connection is locked.
- **onunlock**. A given connection is unlocked.
- **oncommand**. The player types a specific sentence.
- **onwield**. The player wields a given weapon or thing.
- **onunwield**. The player unwields a given weapon or thing.
- **onignite**. A given thing or monster catches fire.
- **onopen**. A given thing is opened.
- **onclose**. A given thing is closed.
- **onenter**. The player enters a given room.
- **onexit**. The player exits a given room.
- **onjump**. The player jumps.
- **onsmell**. The player types **SMELL** in a given room.
- **onlisten**. The player types **LISTEN** in a given room.
- **onturn**. The game reaches a given turn.
- **oneveryturn**. The game executes each and every turn.
- **onputdown**. The player puts down a given thing.
- **onpickup**. The player picks up a given thing.
- **onwear**. The player wears a given piece of clothing.
- **onunwear**. The player removes a given piece of clothing.
- **onpoison**. A given monster is poisoned.
- **onscoreabove**. The player's score is above a certain amount.
- **oneat**. The player eats a given edible thing.
- **ondrink**. The player drinks a given drinkable thing.
- **onkill**. The player kills a given monster.
- **onhpbelow**. A given monster's hitpoints drop below a given value.
- **onexamine**. The player examines a given thing.
- **onhit**. The player hits a given monster.

Events

Once you've got your conditions and your triggers, it's the countdown's events that make things happen.

AAS supports the following events.

- **move**. Moves a given thing or monster to a new location.
- **movetosameroomas**. Moves a given thing or monster to the same location as another thing or monster.

- **enrage**. Makes a monster more ready to attack by shortening the number of turns before it will attack the player.
- **placate**. Makes a monster less ready to attack by extending the number of turns before it will attack the player.
- **conjure**. Makes an invisible thing or monster visible.
- **vanish**. Makes a visible thing or monster invisible.
- **changesound**. Changes the sound of a room.
- **changesmell**. Changes the smell of a room.
- **hurt**. Takes hitpoints away from a monster.
- **heal**. Gives more hitpoints to a monster.
- **ignite**. Sets fire to a thing or monster.
- **douse**. Douses a given thing or monster that's on fire.
- **poison**. Poisons a monster.
- **esuna**. Unpoisons a monster.
- **triggercountdown**. Triggers another countdown, regardless of that countdown's triggers.
- **changehitchance**. Alters a monster's chance of hitting an opponent.
- **changedodgechance**. Alters a monster's chance of dodging an attack.
- **alert**. Pops up an alert box with text that the user can specify.
- **lock**. Locks a connection.
- **unlock**. Unlocks a connection.
- **desc**. Prints some text, presumably describing the event that's happening.
- **destroy**. Destroys a thing.
- **deadly**. Kills the player.
- **changedesc**. Changes the description of a room, monster, or thing.
- **descnear**. Prints some text if the given room, thing, or monster is visible by the player.
- **changeconnection**. Changes a connection's destination to a new room.
- **wear**. Makes a given clothing worn by a given monster.

Advanced Countdown Programming

Some of you who are more used to the intricacies of boolean logic, who are familiar with AND and OR, may be asking, "Roddy, how on God's green earth am I going to get truly complex behaviour out of countdowns?" That's where stringing countdowns together comes in.

Here's what you do. Set up one countdown with the given triggers and some of the conditions you need, either as an AND (all conditions must be true) or OR (at least one condition must be true). Have that countdown's event be a **triggercountdown**. In the second countdown it triggers, have the second part of your logic. You can chain as many countdowns as you need to make this work.

Here's an example. Pretend you have a teleportation device that has a 20% chance of automatically activating every turn. If the player is in the Teleportation Room and the teleporter device's lead shielding has been removed, then the player will be teleported away. However, if the player is holding the Auto-Teleport Transponder, then (s)he will be teleported away regardless of whether or not the lead shielding has been removed.

Here’s how I would create the countdowns required. The first countdown would be triggered every turn with an **oneveryturn** trigger. It would have one condition: an **isprob** of 20%. It would have one event: a **triggercountdown** that would trigger Countdown #2. Countdown #2 wouldn’t need any triggers. It would have two conditions: **isin** and an inverted **isin**. In the first **isin**, I would check to see if the Auto-Teleport Transponder is in the player. In the second **isin**, I would set the inversion flag and see if the Lead Shielding was in the Teleporter thing. I’d also set Countdown #2’s **Conditions needed** to “any,” since either of the conditions can be true. It would have one event: a **triggercountdown** that would trigger Countdown #3. Countdown #3 would have one condition: an **isin** to see if the player is in the Teleporter Room. It would then have one event: a **move** that moved the player to the room where I wanted the Teleporter to teleport the player.

See? Nothing to it. And I’m sure you can image even more complex situations, all of which AAS will handle with ease.

Chapter IV: Other Features

There are a number of other features in AAS that I haven't covered yet.

Verbs

Though AAS contains most of the basic verbs you'll need for your games, you may need to alter the existing verbs or add new ones.

New verbs are more properly synonyms. You could, for example, have a game in which "kick" was the same as "attack." To add new verbs, click on **[New]** in the **New verbs** section.

You can also change verbs. For example, if you're writing an adult text adventure (or XXXTA), you might turn "attack" into "kiss." To change a verb, click on **[New]** in the **Changed verbs** section.

For a full list of verbs AAS supports, see one of the later chapters.

Chapter V: Tips and Techniques

In AAS, there are certain situations that crop up time and again in creating games. I've collected some of them here so that you don't have to re-invent the wheel.

Ending the Game Without Using the Treasure Room

What if you don't want the player to have to end the game by dropping all treasures in the treasure room? No problem. Leave the Treasure Room in the game, but don't have any connections leading to it. Have exactly one treasure thing in the game – you don't have to use treasures to make things that give the player points for getting them – and don't put the thing in any room to start with. When the player is to win, have a countdown that moves the single treasure into the Treasure Room. That will effectively end the game.

Having a Flaming Weapon That Never Burns Up

If you want a flaming weapon (or flaming anything, really) that is never consumed, have a countdown that runs every turn and that adds three hitpoints to the weapon or thing.

Implementing Armour

Have armour increase the player's dodge chance using a countdown that is triggered with an **onwear** trigger. When the player removes the armour, have an **onunwear**-triggered countdown that restores the player's dodge chance.

Implementing Weapons

You can implement weapons in a manner similar to armour. Every time the player wields the weapon, increase his/her attack chance. If you want to be truly fancy, you could also adjust all of the monsters' dodge chances at the same time. When the player unwields the weapon, restore all of the attack and dodge chances.

Keeping Poison From Killing the Player

Normally poison will kill any player or monster eventually. If you want poison that takes off hitpoints for a while, but eventually stops, use a countdown that is triggered when the player eats the poison. Set the countdown's delay to the number of turns you want the player to be poisoned. In the countdown, use an **esuna** event to remove the poison.

Further Tips and Techniques

This is but a small sampling of what you can do in AAS. The best way to learn is to look at other AAS games.

Chapter VI: Thoughts on Text Adventures

I'm more of a toolmaker than a writer, so I don't have a lot to say about text adventures. However, one registered AAS *user, David Banner, has kindly written this chapter. With any luck, his thoughts will help you write better games.*

Text adventures are not the popular mass entertainment that once they were. Some lament this fact, wishing for the money that once fell from on high into the laps of the writers of text adventures. I celebrate this fact. We are freed from such crude considerations to create artistic works that can speak to the ages.

For I do believe that text adventures can be art. But reaching for art does not obviate the need for us to have the craft of text adventures in our hand first. With that in mind, here are some of my musings first on the craft of text adventures, then on the art of them.

Craft

Consider your map carefully. That is, strive to make the layout of your rooms of interest in and of themselves. One of the greatest elements of enjoyment in text adventures is the discovery of new sections of scenery, of new regions of rooms. Bottleneck your rooms: have one room that is the locus between two branches of rooms, so that the player must traverse that room. That way the player will not be able to travel in a straight line and see everything; instead, the player will have to backtrack. Maximum effort can lead to maximum enjoyment.

Manage your mazes. In days of yore, it was sufficient to construct a maze in which connections differed from room to room, but with all of the rooms having the exact same description. Some connections would loop back to the same room. The player would walk north, but could only return to their original room by walking east.

Those days are long gone. Players are tired, yearning for the new. Now you should have a clever solution to the maze. Why make the player wander through a maze, dropping things to map the maze, to give spatial location to what is otherwise a bewildering *tabula rasa* section of the game. Have one clever trick, the application of which will allow the player to breeze through the maze.

Consistency in your craft. In other days of yore, games would be an amalgam of styles and locations, a pastiche of ideas. Dungeons would have cola machines; space colonies, a Victorian parlor. No more. These devices are tired, worn out; try now to hold fast to a central key idea and setting. Consistency may be the hobgoblin of little minds, but it is the kobold, goblin, and troll, but not the cyborg, of the good Dungeons & Dragons-influenced game.

Completeness in your world. If there is something mentioned in your room description, make sure it exists. If your room description includes, say, curtains, then by goodness there should be curtains in the simulation of your world. Use invisible objects if necessary so that the player may look at those curtains. The player should not discover the little man behind the world simulation, which undoubtedly will occur if the player can't look at the things mentioned in a room. Such a jarring occurrence is what I have taken to terming "an unwanted brush with reality."

Art

Text adventures are an odd art form. It is a collaborative one, back and forth, player and author. But there is a third person at our dinner party of text adventures, AAS itself. It is the Greek chorus of our drama, echoing the player's desires and the author's creations.

But beware hubris! As author, you may think yourself an inhabitant of Mount Olympus, but ours is, as I mentioned, a collaborative art. You do not hand down thunderbolts from on high, unless you want your players angry at you. Instead, work with them to weave a tapestry of story whose beauty will astound and amaze all who gaze upon it.

Roddy has said that he is only a toolmaker, not a writer. That is as may be, but toolmakers were artisans of our culture long before the rise of the artist leisure class. The tool Roddy has created will allow us to create art, but that does not remove the art of what he has done. As I have said before, think of our situation as analogous to Plato's cave. AAS is our cave wall; its XML schema, the fire by which we can cast our shadows.

Appendix A: AAS XML Specification

```
<?xml version="1.0" ?>
<game title="(title of the game)" author="(name of author)">
  <maxscore>(integer)</maxscore>

  <room number="(room number)">
    *<bgcolor>(background color for room, default random)</bgcolor>
    <name>(room name)</name>
    *<desc>(room description)</desc>
    *<smelldesc>(smell description)</smelldesc>
    *<sounddesc>(sound description)</sounddesc>
    *<score>(points for getting here)</score>
    *<deadly>(death message)</deadly>
    *<lighted />
  </room>

  <connection number="(connection number)">
    <from>(room number)</from>
    <to>(room number)</to>
    <dir>(direction)</dir>
    *<locked />
    *<key>(thing number)</key>
    *<deadly>(death message)</deadly>
  </connection>

  <thing number="(thing number)">
    <loc>(room number or monster number or thing number)</loc>
    <name>(thing name)</name>
    <noun>(thing noun)</noun>
    *<adj>(thing adjective)</adj>
    *<desc>(thing description)</desc>
    <hp>(hitpoints)</hp>
    *<value>(score for picking up the object)</value>
    *<size>(size from 1 to 100% PC capacity)</size>
    *<weight>(weight from 1 to 100% PC capacity)</weight>
    *<carryease>(how easy to carry from 1 to 100% PC
                 capacity)</carryease>
    *<ammo>(how many times you can use it)</ammo>
    *<outofammo>(message printed when you try to use the thing but
                 it's empty)</outofammo>
    *<FLAG />
  </thing>

  <monster number="(monster number)">
    <loc>(room number)</loc>
    <name>(monster name)</type>
    <noun>(monster noun)</noun>
    *<adj>(monster adjective)</adj>
    *<desc>(monster description</desc>
    <hp>(hitpoints)</hp>
    <hitchance>(percent of this monster hitting opponent)<hitchance>
    <dodgechance>(percent chance of this monster dodging
                  an attack)<dodgechance>
    <ldamage>(min. amount of random damage done)</ldamage>
    <hdamage>(max. amount of random damage done)</hdamage>
    <turnstofight>(number of turns until attack)</turnstofight>
    *<challenge>(phrase shown in turns before attack)</challenge>
    *<attack>(phrase shown when monster attacks)</attack>
    *<death>(phrase shown when monster dies)</death>
    *<missmsg>(phrase shown when monster misses)</missmsg>
    *<dodgemsg>(phrase shown when monster dodges an attack)</dodgemsg>
    *<hitmsg>(phrase shown when monster hits)</hitmsg>
    *<painmsg>(phrase shown when monster loses hp)</painmsg>
    *<value>(score value for beating the monster)</value>
    *<corpse>(thing which is moved into the room when the monster
              is killed, to represent its corpse)</corpse>
    *<FLAG />
  </monster>

  <countdown number="(countdown number)">
    <triggers>
      *<oneveryturn />
      *<onjump />
      *<onexamine target="(object number") />
      *<onsmell target="(object number") />
      *<onlisten target="(object number") />
      *<oncommand>(the specific input line which triggers this
                 countdown)</oncommand>
      *<onenter firstonly="(true|false)" target="(room number)" />
      *<onexit firstonly="(true|false)" target="(room number)" />
      *<onpickup firstonly="(true|false)" target="(thing number)" />
      *<onputdown firstonly="(true|false)" target="(thing number)" />
      *<onopen firstonly="(true|false)" target="(thing number)" />
      *<onclose firstonly="(true|false)" target="(thing number)" />
      *<onturnon firstonly="(true|false)" target="(thing number)" />
      *<onturnoff firstonly="(true|false)" target="(thing number)" />
      *<onwear firstonly="(true|false)" target="(thing number)" />
      *<onunwear firstonly="(true|false)" target="(thing number)" />
      *<onignite firstonly="(true|false)" target="(thing number)" />
      *<ondouse firstonly="(true|false)" target="(thing number)" />
      *<oneat firstonly="(true|false)" target="(thing number)" />
      *<onkill firstonly="(true|false)" target="(monster number)" />
      *<onscoreabove firstonly="(true|false)" score="(number)" />
      *<onhpbelow firstonly="(true|false)" target="(monster number)"
        hp="(number)" />
      *<onturn firstonly="(true|false)" turn="(turn number)" />
      *<onattack firstonly="(true|false)" target="(monster number)" />
      *<onhit firstonly="(true|false)" target="(monster number)" />
      *<onpoison firstonly="(true|false)" target="(monster number)" />
      *<onwield firstonly="(true|false)" target="(thing number)" />
      *<onunwield firstonly="(true|false)" target="(thing number)" />
      *<onlock firstonly="(true|false)"
        connection="(connection number)" />
      *<onunlock firstonly="(true|false)"
        connection="(connection number)" />
    </triggers>
    <conditions logic="(and|or")>
      *<isin not="(true|false)" containee="(thing number)"
        container="(thing number)" />
      *<isvisible not="(true|false)" target="(thing number)"
        from="(object which the given thing needs to be visible
               from)" />
      *<isopen not="(true|false)" target="(thing number)" />
      *<ison not="(true|false)" target="(thing number)" />
      *<iseaten not="(true|false)" target="(thing number)" />
      *<isworn not="(true|false)" target="(thing number)" />
      *<isonfire not="(true|false)" target="(thing number)" />
      *<isdead not="(true|false)" target="(monster number)" />
      *<ispoisoned not="(true|false)" target="(monster number)" />
      *<isprob not="(true|false)" chance="(percent chance of
        event occurring)" />
      *<isscoreabove not="(true|false)" score="(number)" />
    </conditions>
    *<count>(number of times the trigger has to occur before
             the event fires</count>
    *<delay>(number of turns after the trigger occurs before
             the event fires)</loops>
    *<loops>(number of times the event fires)</loops>
    <event>
      *<deadly />
      *<desc>(string to print)</desc>
      *<descnear target="(object where this string will be printed;
        it'll be visible if the PC is nearby)">(event
        description)</descnear>
      *<alert">(text which will appear in an alert box)</alert>
      *<move target="(thing number)" loc="(room or thing number)" />
      *<movetosameroomas target="(thing number)"
        loc="(thing number)" />
      *<destroy target="(thing number)" />
      *<ignite target="(thing number)" />
      *<douse target="(thing number)" />
      *<conjure target="(thing number to make visible)" />
      *<vanish target="(thing number to make invisible)" />
      *<poison target="(thing number)" />
      *<esuna target="(thing number)" />
      *<wear target="(thing number)" target="(monster number)"/>
      *<hurt target="(monster number)" amount="(amount)" />
      *<heal target="(monster number)" amount="(amount)" />
      *<changehitchance minus="(true|false)" target="(monster number)"
        amount="(amount)" />
      *<changedodgechance minus="(true|false)"
        target="(monster number)" amount="(amount)" />
      *<enrage target="(monster number)" amount="(value to subtract
        from turnstofight)" />
      *<placate target="(monster number)" amount="(value to add to
        turnstofight)" />
      *<changedesc target="(number)">(new description)</changedesc>
      *<changesmell target="(number)">(new smell
        description)</changedesc>
      *<changesound target="(number)">(new sound
        description)</changedesc>
      *<triggercountdown>(countdown number)</triggercountdown>
      *<changeconnection connection="(number)" to="(new value for
        destination)" />
      *<lock connection="(number)" />
      *<unlock connection="(number)" />
    </event>
  </countdown>
  <newverb oldverb="(verb)" newverb="(verb)" />

  <changeverb oldverb="(verb)" newverb="(verb)" />

  *<winmessage>(message that's printed when you win)</winmessage>
  *<losemessage>(message that's printed when you lose)</losemessage>

  *<instructions>(instructions)</instructions>
</game>
```

(* designates an optional tag)

Flags

- pickupable
- visible
- treasure
- flammable
- inflammable
- onfire
- poisonous
- weapon
- food
- drink
- clothing
- worn
- table
- box
- open
- turnonable
- on

Appendix B: AAS Verbs

- north (n)
- south (s)
- east (e)
- west (w)
- up (u)
- down (d)
- look (l, examine, x)
- quit (q)
- jump (j)
- pickup (pu, get, take)
- putdown (pd, drop)
- eat
- drink
- wear (don, puton)
- unwear (doff, takeoff, strip, disrobe)
- smell (sniff)
- listen (hear)
- wield (aim, cock, raise, heft, grasp, use)
- unwield (uncock, lower, stow)
- kill (fight, attack, hit, destroy, murder, maim, ruin, break)
- lock
- unlock
- turnon (on)
- turnoff (off)
- open (o)
- close (c)
- brief
- verbose
- semiverbose