ADVENTURE GAME PLANNING

Adventure games have been a popular source of amusement for years. This type of game puts you into a created fantasy world where you make the decisions. For example, you could be a knight in shining armour attempting to rescue a princess from a tall tower guarded by a fire-breathing dragon. In adventure games, the player would be told where he is and what's going on around him, with people and creatures to communicate with, and puzzles to solve. A world where your dreams become reality. The first adventure games were 'text based', meaning that you communicated to the game by typing words and it answered back by printing a reply on the screen. For example, typing "get the lamp" will result in the computer replying "You take the lamp" – providing there was a lamp there in the first place. Playing an adventure game is like being a character in a film, what happens depends on your choice.

The only problem with these early adventure games was that they couldn't understand everything the player typed in. So along came the graphic adventure games which hands control over to the mouse with little or no keyboard input. Commands would be clicked on instead of typed. These types of games include the early Monkey Island and Simon The Sorcerer series. These games bring your adventure to life as you can see the world you are in rather than reading a text description.

PLANNING A GAME

Before we can start writing an adventure game, we need to plan it out. First, we need an idea then built the game around it. Let's say our game takes place in a castle. This would be our fantasy game world which is broken down into several small locations. We would then make a list of the various locations you would find in a castle like this:

Throne Room

Main Hall

Drawbridge

Servants Quarters

Bed Chamber

Tower

Dungeon

Castle Grounds

Staircase

The next thing to do is connect each location to the nearest one. For example, the nearest location to the Drawbridge would be the Main Hall or the Castle Grounds. This is important because you don’t want the player to enter the castle and find himself stepping into the Dungeon. The list would now look like this.

Location 1: Castle Grounds

Location 2: Drawbridge

Location 3: Main Hall

Location 4: Throne location

Location 5: Dungeon

Location 6: Servant’s Quarters

Location 7: Staircase

Location 8: Bed Chamber

Location 9: Tower

OBJECTS

An object is something the player can use in his quest. It could be a coat, a lamp, some food, a bag or anything that can be picked up and moved. Each object has its normal location number, it could also be in special locations which are known as Worn, Carried and Not Created. If the main character is a Barbarian, we could give him a sword and put it in the carried location, so when the game starts the Barbarian is carrying the sword. We could also define a magic potion for him and put it in a normal location waiting for him to find it.

Sword (Carried Location)

Potion (Location Five)

Robe (Worn Location)

If you’ve ever played an adventure game and found something which was hidden, then you’ve found a “Not Created” object. This is a normal object without a normal location number. When the player finds the object the object location variable will become the number of the location the player is in.

PUZZLES

In adventure games, puzzles are there to give the player a challenge and make solving his quest harder. For example: finding a way or opening a chest or getting past a guard dog. Let’s note down some puzzles for the castle game.

Puzzle: The player has to get past the guard at the drawbridge.

Answer: The player must find a guard’s uniform and wear it.

Puzzle: The player can’t unlock the door of the Bed Chamber.

Answer: The player must break the lock with his sword.

Puzzle: The player can’t get past the guard dog in the Tower.

Answer: The player must find a bone and give it to the dog.

TRAPS

As with puzzles, traps are set up for the player to fall into. If he tried attacking the guard then the guard would kill him and the game would be over. Here are some traps for our castle game.

If the player takes the King's Robe the guards will kill him.

If the player drinks the green potion he will die.

If the player tries to walk past the dog it will bite him.

EVENTS

Events are the outcome of things that happen in the game. For example, if the player had solved the puzzle of the dog, then an event would be called to put the bone in the Not Created location and the player would be told that the dog ran off with the bone. The dog and bone would be removed from the game. Here are some more examples of events

When the player examines the vase, he finds the key.

(Note: put the key in the player's present location).

When the player breaks the lock on the door, he can open it.

(Note: set the DOOR variable to one to indicate it’s open).

When the player drinks the potion, his strength will be restored.

(Note: Set STREN variable to full and put the potion in Not Created location).

EXITS (CONNECTIONS)

The player needs a way of moving between locations so we have to define the exits from each location. Exits take on the form of North, South, East, West, Up, Down, etc. We can also have false exits. This means that the player can't move to another location just yet, in other words, it's blocked off. Let's take our castle game. From the drawbridge we have two exits: one leads to the Main Hall (location 3) and the other leads to the Castle Grounds (location 1). We could make a note like this.

Drawbridge: (North goes to location 3, South goes to location 1)

So, the drawbridge is location 2 and from there the player can go north to location 3 or South to location 1. A false exit could be noted down as an event, so when the player opens the door of the dungeon, he can enter it. (Note: Connect Main Hall to Dungeon). You can think of this as a Not Created location being created.

Before we can get these options working, we need to set up the game data. The best way of doing this is to type it all out as data statements and then read them into an array. So, let's put our location descriptions down as data statements first.

```
Data "You are in a cave, an opening is east."
```

```
Data "You are outside a cave, the entrance lies west."
```

To put and hold this in memory we need to define an array using the DIM command like so.

```
dim location$(2)
```

This array will hold two location descriptions, next we need a line to put the location descriptions in the array so we use the READ command.

```
for X=1 to 2 : read location$(X) : next x
```

You can check if the line has done its job by typing 'print location$(1)', the description for location one should appear.

Now let's define the EXIT data. For this, we need to use a two-dimension array called MAP.

```
dim MAP(2,4)
```

Next, we use a nested loop to read the data into the array.

```
for X=1 to 2 : for Y=1 to 4 : read MAP(X,Y) : next Y : next X
```

Finally, the data lines for each location.

```
data 0,0,0,2

data 0,0,1,0
```

Here we have a two-dimension array that holds all the exit data for each location. So, in the above example, the first number is the number of locations in the game and the second is the number of exits.

To use this example we have to give each exit a number for the game to refer to like this.

North – Exit One

South – Exit Two

West – Exit Three

East – Exit Four

So, if we wanted exit two (south) to lead to location 3 we would use the following line of data.

```
data 0,3,0,0
```

Here we have four numbers on a line. As the south is exit two then we replace the second number on the line with the number three so we now have an exit leading south to location three. Let's connect a north exit to location four from this location.

```
data 4,3,0,0
```

Each data statement must have four numbers on it. If you didn't want to use a certain exit then you would set the exit number to zero.

We also need a routine to allow the player to move from location to location via the defined exits. Use this line.

```
if MAP(location,CH)<>0 then location=MAP(location,CH) else print"You can't go that way."
```

OBJECTS

We can use the same method to store the object examine messages.

```
dim OBJECT$(2)for X=1 to 2 : read OBJECT$(X) : next X
```

```
data ”a small lamp.”,”a sharp sword.”
```

OBJECT LOCATIONS

Each object has its own location number which could either be a normal or a special one. First, we set up the array which holds the object location numbers.

```
dim OB_LOC(4)
```

Next, we use READ to put it into the array.

```
for X=1 to 4 : read OB_LOC(X) : next X
```

And next, the data line containing the object location numbers.

```
data 1,5,10,15
```

So, object 1 at location 1, object 2 at location 5, object 3 at location 10, and object 4 at location 15

```
for x=1 to 4

if OB_LOC(X)=location then print”You can see”;OBJECT$(X)

next X
```

Where the OBJECT$ array holds the description of the objects IE: a small lamp.

SPECIAL OBJECT LOCATIONS

Normal object locations would be as many as the locations in the adventure game as they appear in the actual game. But there are special object locations such as these.

CARRIED - Object is carried by the player

WORN - Object is worn by the player

NOT CREATED - Object does not yet exist in the game

To keep them separate from normal objects we need to give them higher numbers than the total number of locations in the game.

CARRIED=1000 : WORN=2000 : NC=3000

Each variable can be used to specify which special location the object is in. So, for example, if we wanted the game to list our carried objects then it could be checked like this. First, using an array like OB_LOC, we can put our first object in the CARRIED special location like this.

OB_LOC(1)=CARRIED

We can then add some lines to our game to inform the player what objects he is carrying……

for X=1 to 10

if OB_LOC(X)=CARRIED then print”You are carrying…..”;OBJECT$(X)

if OB_LOC(X)=WORN then print”You are wearing….”;OBJECT$(X)

next X

I have only covered a few basics here. For full information why not try my STOS Adventure Creator and STOS Graphic Adventure Creator which you can download from the Atari ST section. Most of the work is done for you so you can start knocking out many adventure games.