

Adventure Web

Adventure Web is an interactive adventure game system that allows graphic adventures to be played on the World Wide Web. Release 1.0 of the system is designed as a single player experience.

Each adventure location is defined by a separate Web page. Doors connect to other Adventure Web pages and can span across environments created by many different people. Players can also select their on screen character from African American or white male or female characters.

The system is based on the Jaspar 1 game engine as used in Flight of the Amazon Queen.

Adventure Web Central Station

Clicking of the Gee Whiz! Entertainment logo on the Web Adventure Player will take users to the Gee Whiz! Central Station. This is basically like a train station with lots of doors and where they lead to written above them (eg. Virgin Interactive Shopping Mall, Derek's Monster Dungeon, Tonia's Comic Zone). Every Adventure Web builder is encouraged to submit their URL to Gee Whiz! so that their adventure rooms can be added to the station. All the user has to do is walk through the door to enter someone's adventure room.

Adventure Player and Editor

There are two components to the Adventure Web. These are the player (JASPARW - John and Steve's Programmable Adventure Resource for the Web) and the editor (JOKERW - Jaspar Object Kernel Editor Resource for the Web).

JOKERW needs to be as easy to use as a standard Web page editor. The target Adventure Web builders are people who want to add something different to their web site, yet may have no programming or game design experience.

The Adventure Player records the player's first and last name (%fname, %sname) and their gender (%gender). This can then be passed to the text in the Web Rooms.

How Is Money Generated?

The actual Web player is given away for free, and a version of the Adventure Web Construction Kit containing a Web Editor and User Defined Sprite Editor is also available free from the Gee Whiz! web site for a 30 day evaluation.

Non commercial web sites are free to create Web Adventure pages but are encouraged to register their copy of the kit. The non-commercial version does not contain the User Defined Sprite Editor.

1) Commercial Site License

Commercial sites will pay us a fee to use the Web Adventure format

2) Registration of Web Editor

Non commercial users can register their web editor and receive support

3) Adventure Web Construction Kit

Contains a set of background graphics, room objects and items that can be used to create a number of different and varied Web Adventure pages. Designed specifically for people who cannot draw. Also contains a User Defined Sprite Editor.

Money is also generated by selling new Room sprite and tile sets. There are two sprite bank formats - one that is used in the editor and another that is used in the player. The player sprite set is distributable free of charge.

- These could include space ship, dungeon, shopping centre, village, cave background styles.

What Data Is Loaded From A Page?

- 1) Background tile data with reference to the relevant sprite bank
 - Release2 will be able to load full GIF or JPEG images
- 2) Sprites that animate on that background
- 3) Object data - includes name, description, XY coords, etc.
- 4) Cut scene data
- 5) Dialog data

The background tile data constructs a full screen background out of a few bytes, as opposed to having to load a complete JPEG or GIF image. If the room is constructed from a new background tile set that the user does not have, the relevant tile set bank is transferred from the Web Room site to the users computer.

This data then needs to be checked against a Web Room profile and updated according to previous actions carried out on that page (basically this is an ongoing "save game" feature). This way, if a player picks up a certain object from the screen and it is deleted, then when the player returns, that object is not redisplayed.

Data can be either text based or compressed binary data.

TEXT BASED - has the advantage that people without an Editor can simply modify and create their own "crude" pages.

Stuff That Needs To Be Known Universally

Item Numbers and Game States need to be unique and known universally so that players can visit other sites and build up a list of objects and experience (recorded in Game States).

How do we do this?

- 1) A unique number may be generated using the URL name and the item's number for that room. However, if the web page changes address, the unique numbers are rendered useless.

OR

- 2) Unique numbers are generated by the web authors user name (eg. binary@ecn.net.au) combined with the rooms name. This ensures unique numbers are generated for each page even if authored by the same user. Now, what algorithm do we use to generate these numbers...? Perhaps we can use the decimal notation for the URL?

Game State Data

nr	<i>integer</i>	Unique Game State number
value	<i>integer</i>	Unique value
name	<i>string</i>	Game State name

Web Room Data

nr	<i>integer</i>	Unique room number
webaddress	<i>string</i>	URL address
name	<i>string</i>	Room name
tilebank	<i>integer</i>	Index to tile bank
roomtiledata	<i>tiledata</i>	Contains array for background tile sprites

Area Data

Area data occupies zone plane 2.

nr	<i>integer</i>	Area number
box	<i>boxtype</i>	(TX,TY,BX,BY) coordinates to zone object
fscale	<i>integer</i>	Maximum % size of actors in this area
bscale	<i>integer</i>	Minimum % size of actors in this area
enterscene	<i>index to scene</i>	Calls this scene when actor enters area
timerscene	<i>index to scene</i>	Calls this scene when timer=0
timer	<i>integer</i>	Counts down when hero stands in area
resettimer	<i>boolean</i>	Timer resets when hero enters area

NOTE:

- 1) When placing object sprites on screen and zoning in their boxes - let's do something different to how we did things in Amazon Queen. Let's place all sprites on screen, then zone in an object's box around the sprite and connect it to the object. Now the sprite will only appear if the connecting object's ACTIVE field is true.
- 2) Area boxes occupy a separate zoning field to objects. Object are on plane 1, Areas on plane 2. This allows us to overlap object and area boxes with little conflict. Objects always have priority.
- 3) Have Actors as a predefined set of sprites, eg. Man, Woman, Boy, etc. that are available from sprite bank. This allows people to simply add an actor on screen and the underlying data is not available for edit.

Web Object Data

Object data occupies zone plane 1.

nr	<i>integer</i>	Unique object key
type	<i>objtype</i>	(object,actor)
name	<i>string</i>	Object, actor name
descr	<i>string</i>	Description of object
walkXY	<i>xytype</i>	X,Y coords for HERO to walk to
walkoffXY	<i>xytype</i>	X,Y coords off screen for HERO to walk to
exitobject	<i>objectnr</i>	Object this object connects to if it is an exit

exitstatus	<i>exittype</i>	(nodoor,closed,open,locked)
active	<i>boolean</i>	TRUE = visible, FALSE = hidden
box	<i>boxtype</i>	(TX,TY,BX,BY) coordinates to zone object
pickup	<i>boolean</i>	TRUE = Can pick up this object
usecontext	<i>usetype</i>	(on,with,in) Command constructor referred from target object
uselevel	<i>numtype</i>	Action the object now (singular) or with another object (plural)?
direction	<i>dirtype</i>	(front,back,left,right) HERO faces this way
number	<i>numtype</i>	One object (singular) or group (plural)?
sex	<i>sextype</i>	(male,female,neuter)
take	<i>boolean</i>	Actor only. (TRUE=take , FALSE=use)
image	<i>index to sprites</i>	Points to sprite on screen (0 means no sprite)

Web Sprite Data

nr	<i>integer</i>	Unique sprite number
sframe	<i>integer</i>	Start frame
eframe	<i>integer</i>	End frame (optional)
speed	<i>integer</i>	Animation speed (optional)
scale	<i>integer (1-100)</i>	Size that sprite appears (default 100)
flip	<i>boolean (0,1)</i>	Whether sprite is facing left or right
bank	<i>integer</i>	Index to sprite bank - default is room bank

Web Item Data

IDEA: As each new item is found, save the sprite image to a personalized item sprite bank. This saves having to keep lots of different sprite banks in memory for each item from each possible Adventure Room that a player visits. Simply record the unique Item index and frame in the ongoing save game data.

nr	<i>integer</i>	Unique object key
name	<i>string</i>	Object, Item, Actor name
descr	<i>string</i>	Description of object
give	<i>boolean</i>	(TRUE=give, FALSE=use)
active	<i>boolean</i>	Whether the Item is in inventory or not
usecontext	<i>usetype</i>	(on,with,in) Command constructor referred from target object
uselevel	<i>numtype</i>	Action the object now (singular) or with another object (plural)?
number	<i>numtype</i>	One object (singular) or group (plural)?
sex	<i>sextype</i>	(male,female,neuter)
image	<i>index to graphic/anim</i>	Graphic representation includes anims. (This could possibly be an

index to a personalized item sprite bank)

Web Text Data

text	<i>string</i>	Something to say on screen
XY	<i>xytype</i>	Position to place text
color	<i>integer</i>	Colour of the text
outline	<i>boolean</i>	Does the text have a black outline
width	<i>integer</i>	How wide before word wrapping begins
justification	<i>integer</i>	0 - Left, 1 - Centered, 2 - Right

Dialog Data

text	<i>array of strings</i>	All the options and responses
treedata	<i>array of integers</i>	The dialog display data

Dynalum Data

Save for Release 2.

Actors

Actors can't walk across screens. Only the hero can walk.
Simple talk frames are needed for actors.

Types

objtype	(object,item,actor)
xytype	(X,Y)
exittype	(nodoor,closed,open,locked)
boxtype	(TX,TY,BX,BY)
usetype	(on,with,in)
numtype	(singular, plural)
dirtype	(front,back,left,right)
sexttype	(male,female,neuter)
objectnr	Object this object connects to if it is an exit
tiledata	Background sprite tile data (probably a string)

Talk Codes

These apply only to the hero.

Talk codes are imbedded in descriptions and dialog strings and are translated into a corresponding talk action as a speak string is spoken. They take the form of:

TALKCODE

Eg.

"Hello there...*SMILE*Is that your ship?*POINTBACK*"

Jack smiles as he says hello, then points over his shoulder as he asks about her ship.

<u>TALKCODE</u>	<u>ACTION</u>
SMILE	- Smile
HIPS	- Hands on hips
SCRATCH	- Scratch chin

POINTBACK - Point back
 CROSSARMS - Cross arms
 GESTURE - Gesture with one hand

Default Talk Code is hands by the side.

Player Variables

These are set by the player in the Web Adventure Player program.

%usertitle - Player's title. Can be anything, eg. Mister, Ms., Lord, etc.
 %fname - First name, eg. "John"
 %sname - Surname, eg. "Smith"
 %gender - Player's gender. Is displayed as either "man" or "woman"
 %oppgender - The opposite sex. Derived logically from %gender

Eg.

"Welcome to my shop %fname. I have everything that a young %gender would need."

becomes:

"Welcome to my shop John. I have everything that a young man would need."

Gee Whiz! Sprite Banks

There are two sprite bank formats: one can be loaded into the editor, while the other is freely distributable but can only be loaded by the player. This allows people to purchase extra sprite bank formats from Gee Whiz! to build new Web Adventure rooms and retain the ability to let other people who haven't bought that sprite bank be able to look at their rooms.

Each sprite bank has two files:

- 1) Actual sprite frames
- 2) Index of sprite frames
 - This contains the background tile start and end frames, items, the number of actors and each of their start and end frames as well as predefined animated frames - all with names.

User Defined Sprite Banks

User defined sprite banks allows builders to create a custom sprite bank so that they can add special objects to their own rooms.

Object Commands

Each object has a command list that calls the required scene.

There are three action types:

- 1 **Use** (includes Give/Pick Up)
- 2 **Look At**
- 3 **Walk To**

<u>Action</u>	<u>Obj1 Type</u>	<u>Obj2 Type</u>	<u>Give/Use</u>	<u>Command String</u>
1	actor	nil	nil	TALK TO obj1.name
	object	nil	nil	USE obj1.name

item	object/item	nil	USE obj1.name (obj2.usecontext)
obj2.name			
item	actor	FALSE*	USE obj1.name (obj2.usecontext)
obj2.name			
item	actor	TRUE*	GIVE obj1.name TO obj2.name

*See GIVE and TAKE fields in object data ITEM and ACTOR data definitions

2	all	LOOK AT obj1.name
3	all	WALK TO obj1.name

Give and Take examples:

1) Knife has GIVE=FALSE (it is Use item), Pyxel has TAKE=TRUE (she Takes items), Black Hole has TAKE=FALSE (he uses Items). Use Knife on Pyxel (FALSE | TRUE = TRUE) which is GIVE. Use knife on Black Hole (FALSE | FALSE = FALSE) which is USE.

2) Money has GIVE=TRUE (it is a Give item). Use money on Pyxel (TRUE | TRUE = TRUE) which is GIVE. Use money on Black Hole (TRUE | FALSE = TRUE) which is GIVE.

Table of Mouse Key Commands:

Com = partially constructed command, eg. Use COMOBJECT on..., Give COMOBJECT to...

Empty = An walk to area or a part of screen with object or item.

NOTE: The **pick up** command is implemented in an *object use* script and is called by either a LMB+RMB or LMB+LMB on an object.

	<u>LMB</u>	<u>RMB</u>	<u>LMB+RMB</u>	<u>LMB+LMB</u>
Object (Object)	Walk to object	Look at	Use object	Use object
Object (Item)	Select for use	Look at	Use item	Use item
Object (Actor)	Walk to actor	Look at	Talk to actor	Talk to actor
Empty	Walk to nearest xy	Walk	Walk	Walk
Com + Object	Use on object	Use	Use	Use
Com + Item	Use on item	Use	Use	Use
Com + Com Item	Put item away	Put away	Put away	Put away
Com + Actor	Use/Give to actor	Use/Give	Use/Give	Use/Give
Com + Empty	Reset, walk	Reset, walk	Reset,walk	Reset, walk

Cutaways

- This system is designed so that anyone can construct an adventure page in as little time as possible, so simplicity is required
- Rather than using a script system, it would be easier to use a simple editor similar to the Amazon Queen ACE editor but included in the JOKERW editor
- Only the player's character can move on screen
- Cutaways are attached to objects, items and areas (if attached to Items, they will need to be saved with the Item Sprites in the user save game).

For each cutaway step the following can be done:

- Move hero to X,Y coords
- Hero speaks TEXT
- Actor speaks TEXT
- Place text at X,Y coords in a selected COLOR
- Object graphic image changes
- Object turned on or off
- Item is inserted or deleted
- Pause for N seconds
- Jump to a new ROOM

(possibly allow conditional check for each step, with the ability to follow through for all steps)

Command List

comnum	<i>integer</i>	Unique command number
comobject	<i>objectnr</i>	Refers to the object that calls the command
comaction	<i>actiontype</i>	(use, lookat, walkto) Action type
comtarget	<i>objectnr</i>	Refers to target object
comscene	<i>index to scene</i>	The scene number to call if all of the above conditions are satisfied

NOTE:

comaction matches the action value of the current command.

Why does *look at* have a separate action number?

Although we're calling a script which could just as easily be defined as a *use* script, *look at* is defined as a different action because an object can be *used*, *walked to* or *looked at* at the same time. Conversely *pick up* is defined as a *use* function because objects cannot be used until they are picked up. *Give* is also defined as a *use* function because giving an object to an actor has been defined as the same as using it on that actor. This is a design decision that limits the desired action of an object on an actor to be predetermined as either *give* or *use*. This limits some puzzle designs, for example you cannot give the Shrink Ray to Pyxel to mind as well as use the Shrink Ray on Pyxel to shrink her.

COMMAND LIST EXAMPLE:

action=1	O(10)	usecontext=on	O(4)
use	knife	on	banana

comobject=10 (knife)
 comaction=1 (use)
 comtarget=4 (banana)
 comscene=54 (script: peel banana)

Dialogs

Two types of Dialogs:

- 1) Four dialog options allow a choice of four more dialog options
 The option may set a GS
 The displaying of options can be restricted depending on GS value
- 2) Binary selection, either yes/no answers (a la Zelda)

EDITOR

Typical steps taken to create a Web Adventure room:

Creating Room and Adding Objects

- 1) Use room editor to select tiles to build room background
- 2) Position object sprites on screen, select image direction and adjust size
- 3) Paint in walk areas on screen, adjust scale for each area (default is 100%)
- 4) Draw boxes around object sprites, connect the sprite to object and fill in name, description and walk to XY coordinates - connect object to another Web Room if it's a door
- 5) Add any necessary text to the room

Creating Puzzles and Cutaways

- 1) Select an Object, Item or Area (this becomes **comobject**)
- 2) Select action that calls cutaway (set **comaction** - either use, lookat or walkto)
- 3) If comaction is "use" then set **comtarget**
- 4) Enter in game states that must be tested
- 5) Enter in game states that will be altered
- 6) Edit data for each step