A Guide To Designing Habitat Regions

by
Chip Morningstar

Lucasfilm Ltd. Games Division
December 15, 1987

Introduction

The **Habitat Region Design Kit** is a package of design aids and information that will enable you to design Habitat regions. The document you are now reading is "the manual". The purpose of this document is to provide you with the instructions and reference material needed by Habitat region designers. With the information and materials provided here, you should have everything you need to create your own designs for places and activities in Habitat. We assume that you are already familiar with the Habitat basics from the player's point of view. This document will give you a peek behind the scenes and explain how it is all put together. (Since the information that follows goes into some detail about Habitat internals, we require that you keep it in the strictest confidence.)

The Contents of The Kit

In addition to this manual, **A Guide To Designing Habitat Regions**, the kit also includes the following pieces:

- Copies of the *Region Design Worksheet*, a form showing the layout of the Habitat graphics screen and the important areas and boundary lines on it. Several copies of the worksheet are provided, but you are free to make more copies if you need them.
- Copies of the *Region Map Worksheet*, a form to aid in laying out the interconnections of multi-region areas. As with the Region Design Worksheet, feel to make as many copies of this as you need.
- Template images of all the various objecs available to you for creating regions. These are printed to the same scale as the worksheet, allowing you to layout region designs accurately using cut-and-paste techniques.

Both the worksheet and the template images are available on a Macintosh diskette as MacDraw images, allowing you to perform the cut-and-paste layout operations without resort to such uglinesses as scissors, tape and copy machines. However, throughout this document we will assume that you are working without benefit of electronic assistance.

The Structure of This Document

The remainder of this document is divided into three major sections. The first section is an explanation of the overall constraints on region design. This section will describe how the Habitat software displays regions and what restrictions this places on what you can do. The second section is a reference guide that describes the peculiarities of all the various different classes of objects available to you, the region designer. The third and final section will explain the materials in the kit. It will describe how you actually use the kit to create region designs. We will illustrate this section with an extended example showing the design of an actual region.

Section I — The Structure of Habitat Regions

The Habitat world is made of *regions*. Each region represents one place that you can visit with your avatar. A region has various pieces of information associated with it, such as what other regions it is connected to and what *objects* it contains. This information, especially the objects, determines what sort of place or scene the region will be.

Objects, in turn, are the basic building blocks that you have to work with in constructing scenes. There are numerous *classes* of objects, each representing one sort of thing that you can encounter in the world. For example, doors, teleport booths, streetlamps and signs are all classes of objects. Less obvious classes of objects include ground, sky and graphic primitives such as trapezoids (from the Commodore 64's point of view, avatars and even the region itself are also objects, but we needn't go into that complication at this point).

Each object has some information associated with it that we call its *properties* or *state information* (the two terms are used pretty much interchangeably). Each object has a class (an indicator of what sort of object it is, e.g., a door or a window), graphic style (which of several alternate images to use to display the object, e.g., a wooden door vs. a glass door), graphic state (how the image is to be displayed, e.g., an open door vs. a closed door), X and Y location (where the object is displayed on the screen), orientation (whether the object is displayed normally or ‘‘mirror’’ reversed), and a color or pattern value (for those parts of the object that may be colored). Some objects may have additional state information whose meaning varies depending on the class of the object. For example, for boxes and other containers have properties that tell whether they are open or closed, locked or unlocked, and what key is to be used to lock or unlock them. On the other hand, a flashlight or floor lamp has a property that tells whether the light is on or off. These sorts of class-specific properties will be explained in greater detail in Section II below.

To design a region, you simply specify the properties of the various objects that form the scene you want. In what Section II we will describe the classes of objects that are available for you to use and explain how they work together.

How A Region Is Drawn: The Habitat Graphic Model

Regions are displayed on the Commodore 64 screen in a graphics window that measures 160 pixels across by 128 pixels high. Objects are placed on the screen simply by giving them (X,Y) coordinate positions in this area (i.e., X-positions in the range 0 to 159 and Y-positions in the range 0 to 127). However, as with nearly everything involving the Commodore 64, there are some complications. This sub-section is devoted to explaining these complications.

The X-positions of all objects are restricted to integer multiples of 4 (this is due to quirks of the Commodore's display memory layout that we exploit to obtain faster animation drawing). Thus, the X-coordinate of an object's location is always a number like 4, 8, 24, or 116 but never a number like 3, 9, 22 or 115. For your convenience, the region design worksheet has guidelines showing the multiple-of-4 boundary lines (see fig. 1). Objects with X-coordinates of 160 or more will be off the right-hand edge of the screen. The origin of an object (the point on the object image that is positioned at the object's X-Y location) is usually at the lower left-hand corner of the image. A few object images, however, have pixels to the left of their origin, so locating an object off the right-hand edge of the screen sometimes makes sense. It is not possible to place objects off the left-hand edge since the left-most X-coordinate is 0.

The Y-positions of objects are not restricted to multiples of 4. However, the Y-coordinate encodes not only the Y-position of an object but whether it is to be displayed in the *foreground* or the *background*. The distinction between foreground and background objects will be explained momentarily. Objects with Y-coordinates in the range from 0 to 127 are displayed at the corresponding locations on the screen, in the background. Objects with Y-coordinates in the range 128 to 255 are displayed at the Y-coordinate minus 128, in the foreground. For example, an object with a Y-coordinate of 137 would be displayed in the foreground at Y-position 9, since 137-128=9. In other words, adding 128 to a background object's Y-coordinate turns it into a foreground object without changing its position on the screen. To understand what this means, we need to explain a little bit about how objects are displayed by the Habitat animation software.

Graphically speaking, Habitat objects may be in either the foreground or the background. Background objects are drawn into a stored picture that is used as a backdrop for the scene on the screen. This backdrop picture is changed only infrequently. Foreground objects, on the other hand, are redrawn with every animation frame. The practical consequences of this are as follows. Background objects may not animate nor may avatars walk behind then nor place other objects behind them. On the other hand, a background scene can be arbitrarily complex without degrading the frame-rate. For this reason, whenever possible, scenic objects should be placed in the background.

There is a second important distinction between background and foreground objects. Foreground objects are drawn in top-to-bottom order (or back-to-front order, depending on how you think about it). That is, objects with higher Y-coordinates are drawn first, and then objects with lower Y-coodinates are drawn on top of them. This creates the illusion of depth with objects eclipsing one another appropriately. Background objects, on the other hand, are drawn bottom-to-top. This turns out to work better for creating backdrops that are simply series of layered images with no depth to them. The difference in drawing order sometimes creates subtle problems getting a region to ‘‘come out right’’ when the animation engine draws things in a different order than common sense might otherwise dictate in a particular situation.

Color: More Commodore 64 Complexity Shows Through

Most classes of objects may be colored or patterned. There are 15 possible patterns and 16 possible colors to choose from. All Habitat graphics are stored on the Commodore 64 as images drawn from a palette of four ‘‘colors’’: pink, black, transparent, and *wildcard*. Pink and black pixels are drawn on the screen in pink or black, respectively (actually, ‘‘pink’’ pixels are drawn in the color that is changed by the **F6** key; this starts out pink but can be adjusted by the user to any of the 16 colors that the Commodore 64 supports). Transparent pixels are not drawn at all; the corresponding pixel on the screen is left untouched and whatever was behind the object at that point shows through. Wildcard pixels are drawn on the screen in a color that depends on the pattern/color property of the object being rendered. Fig. 2 illustrates this. Fig. 2a shows a typical Habitat object image, in this case the body sprayer. Fig. 2b shows a random background onto which the image is going to be drawn. Fig. 2c shows the body sprayer drawn onto the background. Note the way the transparent pixels were handled. Finally, fig. 2d shows a pattern being mapped onto the wildcard pixels.

There are 15 patterns that may be applied to an object. These patterns are predefined by the system and cannot be changed. Each pattern is 4-pixels by 4-pixels and is repeatedly tiled over the area to be filled in. See fig. 2d for an illustration of this process in operation. The pattern is aligned to the screen, not to the object, so that the patterns of adjacent objects will line up. Each of the 16 pixels of a pattern is either pink, black or blue. The first 3 of the 15 predefined patterns are solid blue, solid black and solid pink. The remaining 12 are the various textures you are no doubt familiar with (the exact patterns are illustrated in fig. 3).

Objects that are not patterned may be colored (an object may be either patterned or colored, but not both). Colored objects are drawn with the wildcard pixels in one of the 16 colors that the Commodore 64 can display. However, there are some severe restrictions on the use of color due to the way the Commodore 64’s display memory is organized. The screen is divided up into a series of color cells, each of which is 4-pixels wide by 8-pixels high. All colored pixels drawn inside the boundary of a given color cell *must* be the same color (we have no choice in the matter; that is just the way the machine works). The color cells are indicated on the region design worksheet.

Whenever two objects that are colored different colors overlap or are adjacent, the boundary between them *must* lie along the edges of different color cells. If they do not, the colors will ‘‘bleed’’ and you will get horrible square color blocks sticking out beyond the edges of one of the objects. A corollary to this rule is that any object that can be picked up by an avatar (or moved by a player in any way) *must* be patterned and never colored, since there is no way of predicting what other colored objects it might be set in front of or next to. This is the reason for the highly restricted color schemes found in avatars, their heads, and the various implements that they can carry around. Fig. 4 illustrates the operation of color cells and shows how both properly and improperly overlapped objects effect the display.

Other Limitations

A few other graphic conventions must be followed with regard to the placement of foreground objects.

Each region has a ‘‘horizon line’’ that delimits the maximum height to which an avatar may walk on the screen. Ordinarily this is at Y=32 (this line is highlighted on the region design worksheet). You should not deviate from this standard unless you have a special reason for doing so. For example, the chessboard regions have the horizon set at the top of the screen so that players can walk anywhere on the screen they choose. Wherever the horizon line is set, you should avoid placing foreground objects with Y-positions *above* it.

If you for some reason need to place foreground objects above the horizon line, no foreground object must ever be placed blocking the area in the upper left-hand corner of the screen to the left of X-position 16 and above Y-position 104. This is where the ghost icon is displayed when ghosts are present, and the ghost icon *must* always be visible. A foreground object in this location would cover it up.

Foreground objects that may obstruct avatar walking should not be placed in the area to the left of X-position 16 and below the horizon line. This is the area in which avatars are placed when they de-ghost. It must not be possible to de-ghost into a position that it is impossible to walk away from. Although our primary concern is with blocking movement, we also discourage placing objects in this area that do not obstruct walking but that *do* obstruct vision.

You cannot have more than 128 objects in a region at a time, nor may you have more than 64 objects of any given class at a time. Objects inside closed containers do not count towards this limit, but objects inside open containers, even if the objects themselves are not visible, do. There is also a limit of no more than 16 different styles of head allowed at once (you can have more than 16 heads if some of them are of the same style).

Properties of the Region

Each region has a number of properties that control how it operates. We mentioned the horizon line above, but there are several others that are important. Each of these properties must be specified in order to completely describe a region.

Owner

Each region has an ‘‘owner’’. This is the avatar whose turf the region is. A region can be designated as belonging to a particular individual, or it may be flagged as either *public* or *unassigned*. A public region is nobody’s turf and may be entered by any avatar who can get to it. Most regions are public. An unassigned region is one that is set aside to become someone’s turf someday, as new users enter the system. When an unassigned region is created, it goes on a list that the system keeps of all the currently unassigned regions. When a new avatar is hatched, the system grabs the next region from the unassigned region list and makes it their turf. You may want to designate some regions as unassigned if you are designing a residential area.

Light level

A region’s light level is a number that indicates what degree of illumination is ‘‘natural’’ for the region. A value of 0 indicates that the region is lit by daylight, a value of 1 indicates that the region is lit by its own light sources (inside a public building, for example), and a value of -1 indicates that the region is naturally dark (as in a cave). The Habitat system also maintains a world-wide day/night illumination value that is added to the region’s light level to determine whether the region is light or dark. This value is 1 when it is Habitat ‘‘day’’ and 0 when it is Habitat ‘‘night’’ (note: right now there is no day-night cycle -- it’s always daytime). If the sum of the global day/night value and the region’s natural light level is more than 0, then the region is lit; if not, it is dark. Dark regions are displayed with all colors except blue drawn as black.

In addition, the light level in a region can be modified by artificial light sources. Turning on a light in the region increments the region’s light level by one; turning the light off again decrements it. Thus, regions with a natural light level of -1 are dark all the time, unless illuminated with a torch or flashlight; regions with a natural light level of 0 are dark at night and lit during the day; and regions with a natural light level of 1 (or more) are always lit. Note that you can set a region’s light level to less than -1, making it ‘‘extra dark’’ and requiring the use of more than one light source in order to see. Be careful, however, since by doing this it is easy to confuse the players without their realizing that it is just extra dark.

Orientation

A region has an orientation, or facing, with respect to the Habitat compass. The region's orientation is the direction that is up, or to the back, when the region is displayed on the screen. The possible orientations are, naturally, west, north, east or south. ‘‘Normal’’ orientation, in the absence of any particular reason for a particular facing, is to the west.

Neighbors

A region may connect ‘‘naturally’’ to up to four other regions, one in each of the four compass directions. In addition, doors and other passageways may connect to additional regions. This will be explained in more detail in Section II.

Exit Restrictions

For each of the directions from which it is possible to leave a region, you can set a ‘‘restricted’’ flag. If an exit's restricted flag is set, the system will not allow an avatar to leave via that exit while holding a restricted object. This allows you to place objects that can be picked up and manipulated by an avatar but that can't be removed from a given area. This is used, for example, in the Library to keep people from stealing the books.

Weapons Free Zones

A region may be designated as a *Weapons Free Zone*. In such regions weapons will not work and thus avatars will not be able injure each other with them.

Theft Free Zones

A region may be designated as a *Theft Free Zone*. In such regions the system will not allow avatars to grab things from other avatars' hands.

Name

A region may have a ‘‘name’’. This is the string that is given when the player asks for HELP for the region. The name can be up to 20 characters in length. It should identify the region's location, if you want the player to be able to find this out. If the player is supposed to be lost when in this region, then just leave the name blank.

Avatar Limit

The Habitat system imposes a limit that no more than 6 avatars are allowed in a region at one time. However, you can set the limit lower if for some reason this would be useful to you. A limit of 0 means that only ghosts may enter.

Entry And Exit Actions

It is possible to have the system take special actions when someone enters or leaves a region. This requires us to do custom programming, so the number of such actions that we can afford to set up is severely limited. However, if you can make a good argument for it, we may be able to arrange something. Contact us for details.

Section II — The Habitat Object Set

As we explained above, each object in a region has a set of properties that determine how it will be displayed. Objects may have additional, class-specific properties that determine how they will behave. The purpose of this section is to summarize the properties of the various classes of objects and explain what these properties mean. However, we will begin by describing the properties that are common to *all* classes of objects. Although most of these were described or mentioned in the discussion of region drawing in Section I, we will cover them again here for the sake of completeness and ease of reference. Also, the discussion in Section I focused on the graphic aspects of objects, whereas the material that follows deals with objects as objects.

Class

An object's *class* determines what sort of object it is. Class is an object's most fundamental property, in that everything else about the object is determined by its class. Possible classes include door, flashlight, box, teleport booth, and so on.

Graphic Style

When an object is to be displayed in a region, the Commodore 64 reads an image for the object from the disk. What image it uses depends on the object's class and on its *graphic style*. The class provides a list of one or more graphics that may be displayed for the object, while the graphic style determines which of them to use. For example, there are two styles of box object. The first looks like an ordinary box, while the second looks like a pirate's treasure chest. Both work in exactly the same way, differing only in the way they look on the screen.

Note that the same image may be used in more than one class of object. For example, the image of the plain-looking door is both a style of door and a style of chest (in which capacity it simulates a closet).

Once an object is created, its graphic style is fixed more-or-less permanently. It will not ordinarily change as a result of player action or the behavior of other objects.

Graphic State

X

Y

Orientation

Color/Pattern

Restricted

Each class of object may have properties that are unique to the class. However, there are several groups of important classes that are similar to one another and so are treated uniformly by much of the Habitat software. The important cases of this are are described below:

Magic

The amulet, gemstone, knick-knack, magic staff, magic wand, ring, and switch are or can be magical. Similarly, drugs have effects on avatars and sensors can indicate various conditions. Each of these things has a list of possible actions that it can take from which you can choose. Additional actions are possible but require custom programming.

Containers

The bag, box, chest, garbage can, hole, and safe are all containers that may be open or closed, locked or unlocked. Furthermore, the bed, bureaucrat, chair, couch, countertop, display case, glue, pawn machine, table, vendo front, and vendo inside are also containers, though they are always open. In addition, the door, while not a container, obeys the same open/closed, locked/unlocked rules as containers do, and so can be treated as a container for many purposes. Lockable containers have associated with them a key number, as do keys. A key whose key number matches that of a container will lock or unlock the container.

Books and Paper

The book, plaque and paper objects all have text associated with them. Paper text may be both read and written by avatars, but is limited to one page. Books and plaques may have more than one page but may not be written to. The distinction between books and plaques is minor. Books must be held in order to read them, whereas plaques are mounted permanently on the wall and can be read from anywhere in the region. Once you are reading a book or plaque, however, the operation is the same.

Each page of text consists of exactly 16 lines (which may be blank) of up to 40 characters apiece. It is a good idea to number the pages of a multi-page document. If the document is long, it is also a good idea to provide a table of contents or an index so that readers do not have to wait for each page at 300 baud if what they want to read is not near the beginning.

Buildings and Doors

A building or door can be connected to a particular region. Walking to the building or through the open door will take an avatar to this region. If a the door or building is not connected to any region in particular, it will connect by default to whatever region is adjoining in the ‘‘up’’ direction from the region containing it.

Teleports

Elevators and teleport booths have addresses associated with them that are used when ’porting. Teleport addresses are limited to 20 characters, which must include the area code. Elevator addresses are limited to 15 characters, which must include the buildng code. Area codes and building codes are always separated from the rest of the address by a dash. Area codes are allocated by town, so you must use the area code associated with whatever town your region is going to be placed in. For example, the area code of Populopolis is `pop` while the area code of Quantumgrad will be `qua`. Building codes are allocated by building. The elevators in a given building should share the same building code, since an elevator can only take you to another elevator with the same code. Building codes should be unique across buildings, so that you can’t take an elevator from one building to another.

Plants and Rocks

Plants and rocks may have *mass* values associated with them. The mass may be either 0 or 1. A mass of 1 means that the object is too heavy for an avatar to lift, while a mass of 0 means that it can be picked up and carried.

Bureaucrat-in-a-box

The bureaucrat-in-a-box is a way of representing a special ‘‘front-end’’ to the Oracle. Each bureaucrat has a special purpose, such as selling billboard advertising or handling transfers of ownership of turfs. Bureaucrats are customized by placing a head on them. Any of the standard Habitat heads may be used. Ordinarily, independent region designers will not use the bureaucrat object, but if you are designing a government building of some sort you may.

Choke Machine

The Choke machine ordinarily doesn’t do anything. It just steals your money. However, the image of the Choke machine is also a style of stationary magic item, so you can have a magic Choke machine that can do anything that any other magic item might do.

‘‘Flats’’

An object called a ‘‘flat’’ is what we use to represent ground, sky, and so on. A flat is simply a plain rectangular backdrop designed to be positioned on various parts of the screen. Each flat has a ‘‘flat type’’ that determines how it responds to the **GO** command and thus distinguishes its role in the scene. There are four possible flat types: *ground*, *sky*, *wall* and *impassable*. A**GO** to a ground flat results in the a vatar simply walking to the X-Y location indicated by the cursor. A**GO** to a w all flat takes the avatar to the base of the wall at the X location indicated by the cursor. A**GO** to a sk y flat takes the avatar to the next region. A **GO** to an impassable flat fails instantly, thus you cannot walk to or onto a location on an impassable flat.

‘‘Glue’’

‘‘Glue’’ is a special type of container object. It is completely invisible and transparent. Things which are placed inside a glue object are displayed on the screen but they cannot be picked up or moved using the **GET** command. A glue object can hold up to six items, each of which may be given an individual X-Y location offset from the X-Y location of the glue object itself. Therefore the contents of the glue object may be placed anywhere on the screen. To use glue, simply place the objects that need to be glued wherever you want them to go, and then note that they are glued down.

Holes

Holes are another special class of container. Holes are never locked. However, they can only be opened and closed with a shovel object. Closed holes are invisible and behave in most ways just like ordinary ground. An open hole object displays itself as hole which you can then point to and treat like any other stationary container.

Hot Tub

The hot tub is constructed in two halves. The back portion goes into the background and the front portion goes into the foreground. When the two pieces are positioned correctly with respect to each other they look like a single object. However, an avatar can stand or sit between the two halves and will look like he is in the water.

Signs

There are a variety of different sign images. However, signs come in two basic varieties: short signs and long signs. The only difference is that short signs are limited to text of 10 characters or fewer, while long signs may have up to 40 characters. There are a variety of formatting characters that may be embedded in the text that allow you to control the size and positioning of text *(describe these)*.

Trapezoids and Super Trapezoids

A general-purpose graphic primitive is available for the construction of scenery. This is the trapezoid. A trapezoid turns out to be a good deal more flexible than a mere rectangle without imposing the computational burden of an arbitrary polygon. A trapezoid by be any size up to the entire screen. The trapezoid is drawn with its top and bottom edges horizontal and its sides at whatever angle you choose. It is colored or patterned a single color or pattern (optionally, you may specify that it is to be surrounded with a black border). Although trapezoids are drawn quite quickly, they are still relatively expensive and so you should avoid putting them in the foreground if at all possible. Foreground trapezoids, if you use them, should be few and small.

The ‘‘super trapezoid’’ allows you to paint the trapezoid with a nearly arbitrary pattern. The pattern will be tiled over the surface of the trapezoid just like one of the regular fill patterns. However, you are not limited to a 4-pixel by 4-pixel design. Instead, you may use any size pattern subject to the constraint that the edges are powers of two and their product does not exceed 128. Thus your pattern can be 128x1, 64x2, 32x4, 16x8, or any size smaller that you choose. Each pixel of the pattern may be black, pink, blue or a wildcard color of your choosing.

Vending Machines

Vending machines may hold up to 10 items, each of which may be individually priced. Note that we require that the prices not be lower than that which would be paid for the item by the Habitat pawnshop. A vending machine actually consists of two separate objects, one of which represent the front of the machine and the other of which represents the interior. You don’t really need to worry about this detail except to note that a vending machine counts as two objects towards the limit on the number of objects allowed in a region.