

The Power of Vecset

By Mike Hahn

Copyright © 2008 Vecset.net

Date Last Edited: 29-Oct-2008

Table of Contents

The Power of Vecset	1
<u>Introduction</u>	<u>1</u>
<u>Easy to Learn</u>	<u>1</u>
<u>Powerful</u>	<u>3</u>
<u>Originality</u>	<u>3</u>
<u>Animated User Interface</u>	<u>3</u>
<u>Vecset Business Model</u>	<u>4</u>
<u>Transient vs. Persistent Data</u>	<u>4</u>
<u>Structure of VECSworld</u>	<u>6</u>
<u>Optimistic Assumption</u>	<u>7</u>

The Power of Vecset

Introduction

[[Home](#)]

Vecset is a software tool which enables you to create multiplayer board games, as well as games with 2D animation (and ultimately 3D animation). You can log on to [vecsworld.com](#) and play these games with other Vecset users. Vecset is based on a powerful yet easy to learn, object-oriented scripting language called Vecscript. Non-programmers can create drag-and-drop games, and both novice and advanced programmers can use Vecscript to add functionality to these games.

What sets Vecscript apart from other languages is its dual-syntax capability, which defaults to prefix mode (all operators come before their operands), and also gives you the option of infix mode (binary operators come in-between their operands). When prefix mode is enabled, Vecscript resembles Lisp, but when infix mode is enabled, it looks a lot like Java (Vecscript is based on Java).

Easy to Learn

The Vecscript language is based on Java, but unlike Java, only parentheses are used for grouping (no square brackets or curly brackets), and statements/declarations are separated with semicolons. When in the code editor, the user is always in one of 3 code entry modes: free form, structure editor, and code menu. The question mark (?) is used to enter code menu mode. The Escape key toggles between free form and structure editor modes. Structure Editor mode is (possibly) unavailable when the syntax mode is infix (binary operators come in-between their operands). Syntax mode defaults to prefix (all operators come before their operands), and code entry mode defaults to structure editor.

Structure Mode Commands

A bottom-level token (e.g. a keyword, identifier, operator, or constant) or an entire list is often highlighted. Using the Shift key in conjunction with the Up/Down Arrow keys, it is possible to select more than one token/list at a time.

- **Esc** – toggle between Free Form and Structure Editor modes
- **Up Arrow** - go to previous list element
- **Down Arrow** - go to next list element
- **Left Arrow** - go to parent list
- **Right Arrow** - go to first child element (if none, display text cursor following current bottom-level token)
- **Shift+Up/Down Arrow** - select a range of tokens/lists
- **Printable Char.** - incrementally select valid matching menu item (if any)
- **Backspace** - undo insertion of previous printable char.
- **Delete** - delete current token/list
- **Enter** - display text cursor, insert space after cursor (or insert result of incremental menu selection)
- **Space** - display text cursor, insert space before cursor (or insert result of incremental menu selection)
- **Ctrl+Enter** – if at end of line, append blank line (otherwise break line into 2 lines)

Code Menu Commands

A popup menu above or below text cursor (and including text cursor) is displayed. The contents of this menu include all valid code elements in the context of the text cursor (ignoring anything after the text cursor). If the current menu item refers to a list, the entire list is highlighted (defaults to light gray if background color of whitespace is white).

- **Question Mark** - toggle between Code Menu and Free Form/Structure Editor modes
- **Esc** - show/hide code menu
- **Up Arrow** - move selection up (scroll up after pressing Esc)
- **Down Arrow** - move selection down (scroll down after pressing Esc)
- **Left Arrow** - go to parent code menu
- **Right Arrow** - go to lower-level code menu, if any
- **Enter** - go to lower-level code menu (if none, insert current menu item, go to next menu item, or if none, go to parent code menu)
- **Space** - go to lower-level code menu (if none, insert current menu item, go to next menu item, or if none, go to parent code menu; exit Code Menu mode)
- **Printable Char.** - incrementally select matching menu item
- **Backspace** - undo operation of previous printable char.
- **Page Up** - page up after pressing Esc
- **Page Down** - page down after pressing Esc
- **Shift Arrow** – only used if current menu item is repeated, such as a statement in a block, a declaration, or a parameter in a parameter list
 - **Shift Up Arrow** – select previous instance of current menu item
 - **Shift Down Arrow** - select next instance of current menu item
 - **Shift Left Arrow** – insert above current menu item
 - **Shift Right Arrow** - insert below current menu item
 - **Semicolon** – toggle parent list: multi-line/single-line

Keyboard Aid

This feature eases code entry by enabling the user to enter commonly used characters which are relatively hard to type with more easily-typed characters.

- **Hyphen:** press apostrophe ('). Use the double-quote (") for string literals.
- **Code Menu Mode:** press slash (/). Use the question mark (?) to enter divide-by (/).
- **Parentheses:** press comma (,) for the open parenthesis, and period (.) for the close parenthesis. Use the close parenthesis to enter a decimal point.
- **Hyphen (alternate):** when entering an identifier, hold down the Shift key and while doing that, press a letter key. This will enter a hyphen (-) followed by a lowercase letter.

Keyboard Aid is always disabled inside comments and string literals.

RAD-style program development

The Vecset Code Editor is similar to Visual Basic or Delphi, in which the user selects components from the component palette, drops them on the game window, and uses the Object Inspector to modify their design-time properties. The code editor is used to enter all program code, including event handlers.

Powerful

Vecscript is simple enough for beginner programmers to learn, yet powerful enough for professional game programmers to use as a prototyping tool. When the syntax mode is set to infix, Vecscript code strongly resembles Java (with a touch of Object Pascal added for good measure).

Using multiplayer-enabled game components such as Card (playing card, chess piece, letter tile, etc.), Card-stack, Board-grid, and Table-grid, development of multiplayer board games is well within the grasp of programmers and non-programmers alike.

Vecset includes a Level Editor and a Vector Editor (a "vector" is another name for a static/animated object, which may contain other vectors). The Level Editor lets you create worlds, which are inhabited by both static and animated vectors. Some game genres made possible using Vecset include role-playing games and arcade-style (action games).

Originality

Vecset combines elements of 3 existing Internet destinations: Second Life, BYOND, and Gamerz.net. Second Life is a 3D world that lets its inhabitants do stuff that they can do in the real world, including socializing and playing computer games. BYOND, which stands for Build Your Own Net Dream, lets you play and create multiplayer games. The Gamerz.net game server lets you play multiplayer board games by email.

Like Second Life, Vecset gives you access to a virtual world (called VECSworld), although the graphics are 2D rather than 3D. Unlike Second Life, VECSworld is primarily devoted to gaming. Also, VECSworld lets you create your own games, instead of just playing games others have created.

Like BYOND, Vecset lets you play and create multiplayer games. Unlike BYOND, you don't have to download anything to play games (they run in your web browser), and you don't need any programming skills to create simple games and game prototypes. The scripting language used to create Vecset games (called Vecscript) is at once powerful, easy to learn, and general-purpose in nature, whereas the scripting language included in BYOND is specialized for the development of role-playing games.

Like Gamerz.net, Vecset lets you play multiplayer board games by email. Unlike Gamerz.net, you can also play board games in your web browser (in real-time), and play/create all types of games, not just board games.

Animated User Interface

The VECSworld user interface (when the user is not in a game) is 2-dimensional, consisting of 2 rectangular windows side-by-side (split screen). One window is always an overhead view, and the other window is split into 2 windows, one on top of the other: left-side/right-side, or front/rear. All 3 windows are separated by splitters, so it's easy to resize them. Clicking on the point of intersection in the center of the screen resets the sizes of the 3 windows to their default values.

The user uses the cursor keys (or a joystick) to go forward (up arrow), left, right, or backwards (down arrow). Ctrl+Left Arrow and Ctrl+Right Arrow rotates the overhead

view 90 degrees in the given direction. Clicking on an object carries out the default primary user action, and right-clicking on an object brings up a popup menu of choices. Pressing Tab highlights the next command in an onscreen menu. Clicking the mouse in the non-overhead view toggles between left/right and front/rear.

The default user interface for animated games is similar to that of the VECSworld user interface. Also, right-clicking in the non-overhead view and selecting Backpack (or Ctrl+B) displays contents of backpack in the window clicked upon.

Vecset Business Model

Anyone can log on to vecsworld.com as a guest and explore the streets, buildings, floors, and rooms of VECSworld, as well as kibitz games in progress. To play games, users must first register (which is free), which involves entering their real name and email address, and picking a user name and password. To create games (also free), users must download and install the Vecset IDE (game editor). All game players and game designers are encouraged to pay an annual fee and become members of VECSworld. Game players who choose not to become members face certain restrictions, which are described in the next section.

The main motivator to encourage gamers to become gaming members, thereby coming up with the membership fee of \$24/year, is that every member is assigned a private room, and the data contained therein is preserved even after the member logs off.

The main motivator to encourage game designers to become designer members, thereby coming up with the membership fee of \$48/year, is freedom to locate their games anywhere in VECSworld, instead of the default location depending on the genre of that particular game. Also, they can customize their game rooms any way they want. All game designers who are members are automatically gaming members, so they are also assigned their own private rooms.

Transient vs. Persistent Data

All Vecset data consists of rooms, vectors, lists, atoms, and characters. A character can be either a human player (a user) or a non-player character (NPC). A user can be either a member or a casual user (non-member). A vector is an object, either animated or static, which may contain lists, atoms, or other vectors. An atom is a bottom-level object, such as a polygon, label, bitmap, edit box, radio button, checkbox, or string, which contains no vectors or other atoms. A list is a collection of atoms or other lists. A room is a physical location in VECSworld containing zero or more vectors/characters.

All atoms/vectors are either persistent (retain their value from one game session to the next), or transient (are always initialized to the same value, or chosen at random from a list of atoms at the beginning of a game session). Every time a persistent atom is changed, a time stamp record is added to a database, sorted chronologically. Each record consists of the User ID of the character responsible for the change, the date/time of the change, and the Vector ID of the top-level vector that contains the changed atom. The User ID is zero if the atom-change was game-generated rather than user-generated. Subsequent to a time stamp record being added, the User ID is compared with the Last User ID value associated with the top-level vector. If those values are not equal, and the user type associated with the Last User ID is "member", then the top-level vector is saved to disk. The Last User ID is then set to the User ID.

Game Session Types

All Vecset games are either open-ended or finite. An example of an open-ended game is a Massively Multiplayer Online Role-Playing Game (MMORPG), in which players may log on and log off but the game as a whole continues 24/7. An example of a finite game is a board game or an arcade-style game, in which all players join in at the beginning of the game session and the game session eventually comes to an end for all players.

End of Game Session

At the end of a finite game session, the time stamp database is scanned, including only those records more recent than the date/time of the beginning of the game session. If the top-level vector associated with the time stamp record is transient, then the record is skipped. Otherwise, if the user type of the Last User ID is "member", then the top-level vector is saved to disk.

At the end of an open-ended game session, the time stamp database is scanned, including only those records more recent than the date/time of the beginning of the game session. If the top-level vector associated with the time stamp record is transient, then the record is skipped. Otherwise, if the user type of the Last User ID is "member", then the top-level vector is saved to disk. Otherwise, if the Last User ID is the same as the User ID of the user who is logging off, the value of the top-level vector in RAM reverts to its saved value.

The above scheme for handling persistent data is aimed at making it difficult for rogue game designers to enable casual users to have their own private rooms, which is a perk that should only be enjoyed by members.

Saving the Backpack

Open-ended games which allow users to carry a backpack containing zero or more vectors also allow casual users to retain changes made to any atom contained within their backpacks from one game session to the next. However, the backpack of a casual user becomes transient 30 days after that user first plays a given game. So after 30 days any changes to the casual user's backpack are not saved from one game session to the next.

The data associated with every user (including casual users) includes a 4-byte value which is preserved even after that user logs off. Any game can include Vecscript code to get or set this value.

The above scheme for handling persistent data is designed to allow casual users to play most games relatively freely, while still giving them some incentive to become members. When a game designer downloads and installs the Vecset IDE, and accepts the license agreement, that designer agrees not to design games which attempt to get around this scheme. Hopefully the vast majority of games will not attempt to circumvent this scheme, giving casual users incentive to join and support VECSworld.

Structure of VECSworld

Navigating VECSworld is made easy by its multilevel, grid-oriented structure (when grid mode is enabled). At the top level is the world grid, the next lower level is the city grid, then the block grid, then the building grid, and then the room grid. Each cell in the world grid is a city, each cell in the city grid is a block, each cell in the block is a building (which may have multiple floors), each cell in the building is a room, and each cell in the room (assuming the room is a game room) is a table, with one or more players sitting at that table, either playing a game or waiting for a game to begin. When grid mode is disabled, the user interface of VECSworld is similar to an animated game (see above: Animated User Interface).

Players can use the keyboard or the mouse to navigate through the various levels of grids. The arrow keys move to the cell above, below, to the left, or to the right, like a spreadsheet. Pressing Enter or double-clicking moves to the next lower level grid. Pressing Esc or right-clicking and selecting parent-grid moves to the next higher level grid. At the building level, pressing Ctrl+Up/Ctrl+Down, or right-clicking and selecting Up/Down, moves to the next higher/lower floor.

Every grid is laid out like a spreadsheet, except the cells are usually square rather than rectangular. Each cell address has a column letter followed by a row number. The maximum no. of columns is 26 (A – Z), and there is no maximum no. of rows. Cells in the grid which are not occupied are colored black. Other colors may be used, for instance, at the room level, tables with a game in progress are colored differently than those tables occupied by players waiting for the game to begin.

Optimistic Assumption

Operating under the assumption that VECSworld will eventually rival Second Life in popularity and network capacity requirements, the following 4 paragraphs are an attempt to deal with that situation.

Democracy

VECSworld is run by its members, who can vote in elections in order to elect their leaders. Only members can vote in elections or run as candidates in those elections. The lowest level is the block, the head of which is called a captain. The next higher level is the district (a row of blocks), the head of which is called a councilor. The next higher level is the city, the head of which is called a mayor. The highest level is the world, and the world leader is called the President of VECSworld.

Infrastructure

In case the demand for network traffic exceeds the amount that vecsworld.com is able to supply, vecsworld.com may sell web hosting rights to third parties, who supply web server capacity and receive a portion of the membership fees. Those membership fees are paid by members residing in blocks whose web server capacity is supplied by the third parties.

Helping the Third World

In order to thank the OLPC community for helping in the development of the XO version of Vecset (and possibly helping out with the Java/Windows version of Vecset as well), vecsworld.com pledges to donate one dollar from the sale of every \$24 annual membership fee to One Laptop Per Child. The XO version of Vecset is of course free for all XO users.

Financial Estimate

If Vecset is to be a success financially, the projected user base is let's say 5000 users, at a subscriber rate of say 20 percent, or 1000 members, and let's say an additional 200 game designers who are also members. So the gross income, not including advertising revenue, equals $24 \times 1000 + 48 \times 200 =$ approx. \$34,000 per year. Let's say the average time spent online per user equals 2 hours/week. That's 10,000 user-hours per week. Dividing by $24 \times 7 = 168$ hours in a week = approx. 60 users logged on at any one time.