

# **Vecset Prototypes**

By Mike Hahn

Copyright © 2008 Vecset.net

Date Last Edited: 27-Oct-2008

# Table of Contents

<b>Overview .....</b>	<b>1</b>
<u>Introduction .....</u>	<u>1</u>
<u>Implementation Plan.....</u>	<u>1</u>
<b>Board Game Prototype .....</b>	<b>2</b>
<u>Board Game Classes.....</u>	<u>2</u>
<u>Card.....</u>	<u>2</u>
<u>CardStack.....</u>	<u>3</u>
<u>RowGrid.....</u>	<u>4</u>
<u>BoardGrid.....</u>	<u>4</u>
<b>Animated Game Prototype.....</b>	<b>5</b>
<u>Animated Game Classes.....</u>	<u>5</u>
<u>Card.....</u>	<u>5</u>
<u>CardStack/Vector.....</u>	<u>6</u>
<u>RowGrid.....</u>	<u>7</u>
<u>BoardGrid.....</u>	<u>7</u>
<u>Container Classes.....</u>	<u>8</u>
<u>Animation Events.....</u>	<u>9</u>

# Overview

## Introduction

[ [Home](#) ]

Vecset is used to create multiplayer board games, as well as animated games. You can log on to vecsworld.com and play these games with other Vecset users. The first version of Vecset (the prototype) is implemented in Python, and runs on the XO Laptop. Two prototypes will be implemented: one for board games, and one for animated games.

The difference between the board game prototype and the full board game version is that the prototype board is always a grid, and each cell is always either blank, a circle, a square, or a single character (letter, digit, or punctuation mark). Each cell has 4 colors: foreground, background, border, and pen color. (The pen color is used for the outline of the circle/square. The default cell border can be any color, including none.) All characters on a given board are in the same font/size (there may be more than one board onscreen). The full version supports cells containing arbitrary text and/or bitmaps/vector graphics.

The difference between the animated game prototype and the full animated game version is that the prototype only supports animations of the cells described in the previous paragraph: circles, squares, or single characters. The full version supports animated objects containing arbitrary text and/or bitmaps/vector graphics.

## Implementation Plan

### Outline:

1. Design the board game prototype
2. Design the animated game prototype
3. Advertise on OLPC mailing lists seeking Vecset collaborator
4. Hire collaborator to implement prototypes in Python (for the XO Laptop)
5. Develop Vecscript compiler
6. Develop Java/Windows versions of prototypes
7. Collaborator to expand prototypes to full Python versions
8. Expand Java/Windows prototypes to full versions
9. Develop vecsworld.com, a destination for gamers and game designers

Steps 1 and 2 have recently been performed, and this document contains the design specs. If I am unsuccessful at first in recruiting a collaborator, I will begin developing the Python prototypes myself and then repeat Step 3. Both the collaborator and myself will receive an equal share of the proceeds from subscription fees after vecsworld.com goes live. In Step 5, Vecscript is the built-in scripting language of the Windows version of Vecset, which is based on Java (and also implemented in Java). Steps 4 and 5 are simultaneous, as are Steps 6 and 7.

# Board Game Prototype

## Board Game Classes

The user interface of Vecset board games consists of a top-level window containing one or more board game components, as well as standard widgets such as labels, buttons, and checkboxes. If drag mode is enabled, the user must drag Card objects from one location and drop them on another location. If drag mode is disabled, the user clicks on the source Card object, which is then highlighted, and then clicks on the destination location.

### Drag-and-Drop Components

Drag-and-drop board games are constructed out of 4 basic components: 1) Card, such as a playing card or chess piece; 2) CardStack, a stack of Card objects; 3) BoardGrid, a grid in which each cell is a CardStack object; 4) RowGrid, a collection of CardStack objects arranged in a row (or column). Each of these components has an associated class.

### Board Game Events

A drag and a drop event is generated every time the user drags/drops a Card or CardStack object. If drag mode is disabled, both events are generated when the user clicks on the destination location.

### Standard Widgets

Various widgets the user can interact with, including labels, buttons, checkboxes, radio buttons, edit boxes, memo boxes, combo boxes, group boxes, panels, etc.

## Card

A Card object can be a circle, square, or printable character. A printable character can be a letter, digit, or punctuation mark (ASCII 33 - 126). Every Card object has 4 colors associated with it: foreground, background, border, and pen color. The pen color is used for the outline of the circle/square. The border color is used for the border of the cell containing the Card object. A Card object is always contained within a CardStack object.

### Properties:

**value:** 0 = empty, 1 = circle, 2 = square, 33 – 126 = printable char. (ASCII value); a negative value indicates Card object belongs to opposing player in a 2-player game.

**character:** one-character string (null string if not printable character)

**colors:** foregroundColor, backgroundColor, borderColor, penColor

**dimensions:** width, height (in pixels)

**margins:** leftMargin, topMargin (in pixels); distance from upper left corner of circle/square to upper left corner of cell

**hasBorder:** if true, a rectangular one-pixel wide border is displayed, marking the border of the cell

**transparent:** if true, the circle/square or printable char. is treated as a transparent bitmap.

**canDrag:** if false, user cannot drag away this Card object

**canDrop:** if false, user cannot drop Card objects onto this Card object

## Playing Cards

Printable chars.: AKQJ098765432

Suit colors: spades = black, clubs = gray, hearts = red, diamonds = pink

## CardStack

A CardStack object is a stack of Card objects. Every Card object is contained within a CardStack object.

### Properties:

**containerId:** a unique integer identifying a container of Card objects. If zero, the CardStack object is contained in a RowGrid object.

**cards:** a list of Card objects

**dimensions:** width, height (in pixels)

**position:** left, top (in pixels)

**offsets:** leftOffset, topOffset (in pixels); distance between adjacent Card objects

**margins:** leftMargin, topMargin (in pixels); distance from upper left corner of circle/square to upper left corner of cell

**colors:** backgroundColor, borderColor, selectedColor, selectedForegroundColor

**font:** font of Card objects containing printable characters

**hasBorder:** if true, a rectangular one-pixel wide border is displayed, marking the border of the cell

**destList:** a list of containerId values indicating valid destinations that can accept drag operations originating with this CardStack object

**clickable:** if true, and length of destList equals one, then user need only click on this CardStack object, sending its Card object(s) to destination container object

**canDrag:** if false, user cannot drag Card objects away from this CardStack object

**canDrop:** if false, user cannot drop Card objects onto this CardStack object

**infinite:** if true, top Card object is duplicated rather than moved when user drags Card object away from this CardStack object

**random:** if true, when user drags Card object away from this CardStack, the Card object is selected at random from all Card objects contained in this CardStack. If both random and infinite are true, the selected Card object is duplicated/copied rather than moved.

**visible:** if false, CardStack is not visible to user

**multiDrag:** if true, user can drag more than one Card object away from this CardStack at once. If false, user can only drag away the top Card object.

**isDefault:** if true, all properties of this CardStack are set equal to the properties of the defStack object of the parent RowGrid object.

## RowGrid

A RowGrid object is a row or column of cells, and each cell contains a CardStack object. If the horizontal property is true, the cells are arranged in a row, otherwise the cells are arranged in a column.

### Properties:

**containerId:** a unique integer identifying a container of Card objects.

**defStack:** the default CardStack object

**count:** the no. of cells (CardStack objects)

**cells:** a list of CardStack objects of length equal to the count property

**dimensions:** width, height (in pixels)

**position:** left, top (in pixels)

**offset:** distance in pixels between adjacent CardStack objects (width/height = offset multiplied by count)

**horizontal:** if true, the cells are arranged in a row, otherwise the cells are arranged in a column

**visible:** if false, RowGrid is not visible to user

## BoardGrid

A BoardGrid object is a collection of RowGrid objects. Each RowGrid object's horizontal property is true. Adjacent RowGrid objects are positioned one on top of the other.

### Properties:

**containerId:** a unique integer identifying a container of Card objects.

**defStack:** the default CardStack object

**count:** the no. of RowGrid objects

**rows:** a list of RowGrid objects of length equal to the count property

**dimensions:** width, height (in pixels)

**position:** left, top (in pixels)

**offset:** distance in pixels between adjacent RowGrid objects (height = offset multiplied by count)

**visible:** if false, BoardGrid is not visible to user

# Animated Game Prototype

## Animated Game Classes

The main difference between animated games and board games is that CardStack objects are animated and can move about the screen. The User CardStack usually contains just one Card and is always located in the center of the screen. The arrow keys (or a joystick) are used to scroll up, down, left or right.

### User Interface Commands:

**Move User:** arrow key or joystick

**Primary Action:** left-click

**Secondary Action:** right-click, select from popup menu

**Move Object:** drag and drop

**Zoom Out:** Page Up; zoom in/out by factor of 2

**Zoom In:** Page Down

**Display Entire Level:** Ctrl+Page Up

**Maximum Zoom In:** Ctrl+Page Down

**Show/Hide Backpack:** Ctrl+B

## Card

A Card object can be a circle, square, or printable character. A printable character can be a letter, digit, or punctuation mark (ASCII 33 - 126). Every Card object has 4 colors associated with it: foreground, background, border, and pen color. The pen color is used for the outline of the circle/square. The border color is used for the border of the cell containing the Card object. A Card object is always contained within a CardStack object.

### Properties:

**value:** 0 = empty, 1 = circle, 2 = square, 33 – 126 = printable char. (ASCII value); a negative value indicates Card object belongs to opposing player in a 2-player game.

**character:** one-character string (null string if not printable character)

**colors:** foregroundColor, backgroundColor, borderColor, penColor

**dimensions:** width, height (in pixels)

**margins:** leftMargin, topMargin (in pixels); distance from upper left corner of circle/square to upper left corner of cell

**hasBorder:** if true, a rectangular one-pixel wide border is displayed, marking the border of the cell

**transparent:** if true, the circle/square or printable char. is treated as a transparent bitmap.

**canDrag:** if false, user cannot drag away this Card object

**canDrop:** if false, user cannot drop Card objects onto this Card object

## CardStack/Vector

A CardStack object is a stack of Card objects. Every Card object is contained within a CardStack object.

A Vector object is a CardStack in motion. It includes 2 additional properties: dx/dy, or x/y velocity in pixels/clock-tick. Both of these properties are floating point. Normally a frame change occurs every clock-tick.

### Properties:

**containerId:** a unique integer identifying a container of Card objects. If zero, the CardStack object is contained in a RowGrid object.

**cards:** a list of Card objects

**dimensions:** width, height (in pixels)

**position:** left, top (in pixels)

**offsets:** leftOffset, topOffset (in pixels); distance between adjacent Card objects

**margins:** leftMargin, topMargin (in pixels); distance from upper left corner of circle/square to upper left corner of cell

**colors:** backgroundColor, borderColor, selectedColor, selectedForegroundColor

**font:** font of Card objects containing printable characters

**hasBorder:** if true, a rectangular one-pixel wide border is displayed, marking the border of the cell

**destList:** a list of containerId values indicating valid destinations that can accept drag operations originating with this CardStack object

**clickable:** if true, and length of destList equals one, then user need only click on this CardStack object, sending its Card object(s) to destination container object

**canDrag:** if false, user cannot drag Card objects away from this CardStack object

**canDrop:** if false, user cannot drop Card objects onto this CardStack object

**infinite:** if true, top Card object is duplicated rather than moved when user drags Card object away from this CardStack object

**random:** if true, when user drags Card object away from this CardStack, the Card object is selected at random from all Card objects contained in this CardStack. If both random and infinite are true, the selected Card object is duplicated/copied rather than moved.

**visible:** if false, CardStack is not visible to user

**multiDrag:** if true, user can drag more than one Card object away from this CardStack at once. If false, user can only drag away the top Card object.

**isDefault:** if true, all properties of this CardStack are set equal to the properties of the defStack object of the parent RowGrid object.

## RowGrid

A RowGrid object is a row or column of cells, and each cell contains a CardStack object. If the horizontal property is true, the cells are arranged in a row, otherwise the cells are arranged in a column.

### Properties:

**containerId:** a unique integer identifying a container of Card objects.

**defStack:** the default CardStack object

**count:** the no. of cells (CardStack objects)

**cells:** a list of CardStack objects of length equal to the count property

**dimensions:** width, height (in pixels)

**position:** left, top (in pixels)

**offset:** distance in pixels between adjacent CardStack objects (width/height = offset multiplied by count)

**horizontal:** if true, the cells are arranged in a row, otherwise the cells are arranged in a column

**visible:** if false, RowGrid is not visible to user

## BoardGrid

A BoardGrid object is a collection of RowGrid objects. Each RowGrid object's horizontal property is true. Adjacent RowGrid objects are positioned one on top of the other.

### Properties:

**containerId:** a unique integer identifying a container of Card objects.

**defStack:** the default CardStack object

**count:** the no. of RowGrid objects

**rows:** a list of RowGrid objects of length equal to the count property

**dimensions:** width, height (in pixels)

**position:** left, top (in pixels)

**offset:** distance in pixels between adjacent RowGrid objects (height = offset multiplied by count)

**visible:** if false, BoardGrid is not visible to user

### Navigation:

To move to adjacent cell, press an arrow key. To move a Card to an adjacent cell, click on it and then press an arrow key. To stop moving a Card and start moving the User CardStack again, click on the User CardStack (center of screen). Then press an arrow key.

## **MazeGrid**

A MazeGrid object is similar to a BoardGrid object, except every cell has 0 to 4 visible borders, or walls (default width = 3 pixels). Most CardStacks, including the User CardStack, cannot move between adjacent cells separated by a wall. When a Vector other than the User CardStack moves between adjacent cells, the Vector usually moves smoothly, one pixel or a few pixels at a time, instead of jumping from one cell to another.

## **Container Classes**

### **BackPack**

A list of CardStack objects currently being carried by the user. To show/hide backpack, press Ctrl+B. To cycle through contents of BackPack, press Tab/Shift+Tab. To use currently selected object in BackPack, press Enter (or click on it). To pick up a CardStack object and place it in BackPack, drag it to BackPack.

### **Room**

A Room object is a rectangular region that can contain any or all of the aforementioned objects. Along any of its 4 walls it can contain 0 or more doors, which can be either open or closed.

### **Corridor**

A Corridor object is a long, relatively narrow region with 2 walls on either side, and may have a wall at either end. It may contain doors/openings along any of its walls. The width of a Corridor object is generally a multiple of the default cell width (usually this multiple is 1 or 2). To travel down a Corridor, press an arrow key multiple times (or hold it down). To jump from one side of a vertically oriented Corridor to the other, press the Left/Right Arrow Key (or Up/Down if the orientation of the Corridor is horizontal).

### **Level**

A Level object is a collection of Rooms, Corridors, and other objects. To display the entire Level at once, filling the screen, press Ctrl+Page Up. To return to a normal magnification level, press Ctrl+Page Down. There may be more than one level per game.

## Animation Events

A drag and a drop event is generated every time the user drags/drops a Card or CardStack object. If drag mode is disabled, both events are generated when the user clicks on the destination location.

### Vector Events:

- **OnCreate**
- **OnDestroy**
- **OnShow** – visible changes to yes
- **OnHide** – visible changes to no
- **OnEnter** – enters user's field of view
- **OnExit** – exits user's field of view
- **OnStart** – begins new trajectory, or velocity becomes non-zero
- **OnStop** – ends old trajectory, or velocity becomes zero
- **OnReparent** – new parent object (e.g. enter/exit room)
- **OnAdd** – acquires new Card(s) or CardStack
- **OnSubtract** – loses old Card(s) or CardStack
- **OnCollide** – collision detected
- **OnHitWall** – collision with wall detected
- **OnClick**
- **OnDbClick**
- **OnMouseUp**
- **OnMouseDown**
- **OnMouseMove**
- **OnKeyUp**
- **OnKeyDown**
- **OnKeyPress**