

TileGamer Prototypes

By Mike Hahn

Copyright © 2009 TileGamer.com

Date Last Edited: 19-May-2009

Table of Contents

Overview	1
<u>Introduction</u>	<u>1</u>
Board Game Prototype	2
<u>Board Game Classes</u>	<u>2</u>
<u>Tile</u>	<u>2</u>
<u>Tile-stack</u>	<u>3</u>
<u>Row-grid</u>	<u>4</u>
<u>Board-grid</u>	<u>4</u>
Animated Game Prototype.....	5
<u>Animated Game Classes</u>	<u>5</u>
<u>Tile</u>	<u>5</u>
<u>Tile-stack/Vector</u>	<u>6</u>
<u>Row-grid</u>	<u>7</u>
<u>Board-grid</u>	<u>7</u>
<u>Container Classes.....</u>	<u>8</u>
<u>Animation Events</u>	<u>9</u>

Overview

Introduction

[[Home](#)]

TileGamer is used to create multiplayer board games, as well as animated games. You can log on to TileGamer.com and play these games with other TileGamer users. 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.

Board Game Prototype

Board Game Classes

The user interface of TileGamer 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 Tile objects from one location and drop them on another location. If drag mode is disabled, the user clicks on the source Tile 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) Tile, such as a playing card or chess piece; 2) Tile-stack, a stack of Tile objects; 3) Board-grid, a grid in which each cell is a Tile-stack object; 4) Row-grid, a collection of Tile-stack 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 Tile or Tile-stack 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.

Tile

A Tile object can be a circle, square, or printable character. A printable character can be a letter, digit, or punctuation mark (ASCII 33 - 126). Every Tile 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 Tile object. A Tile object is always contained within a Tile-stack object.

Properties:

value: 0 = empty, 1 = circle, 2 = square, 33 – 126 = printable char. (ASCII value); a negative value indicates Tile 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 Tile object

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

Playing Cards

Printable chars.: AKQJ098765432

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

Tile-stack

A Tile-stack object is a stack of Tile objects. Every Tile object is contained within a Tile-stack object.

Properties:

containerId: a unique integer identifying a container of Tile objects. If zero, the Tile-stack object is contained in a Row-grid object.

cards: a list of Tile objects

dimensions: width, height (in pixels)

position: left, top (in pixels)

offsets: leftOffset, topOffset (in pixels); distance between adjacent Tile 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 Tile 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 Tile-stack object

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

canDrag: if false, user cannot drag Tile objects away from this Tile-stack object

canDrop: if false, user cannot drop Tile objects onto this Tile-stack object

infinite: if true, top Tile object is duplicated rather than moved when user drags Tile object away from this Tile-stack object

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

visible: if false, Tile-stack is not visible to user

multiDrag: if true, user can drag more than one Tile object away from this Tile-stack at once. If false, user can only drag away the top Tile object.

isDefault: if true, all properties of this Tile-stack are set equal to the properties of the defStack object of the parent Row-grid object.

Row-grid

A Row-grid object is a row or column of cells, and each cell contains a Tile-stack 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 Tile objects.

defStack: the default Tile-stack object

count: the no. of cells (Tile-stack objects)

cells: a list of Tile-stack objects of length equal to the count property

dimensions: width, height (in pixels)

position: left, top (in pixels)

offset: distance in pixels between adjacent Tile-stack 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, Row-grid is not visible to user

Board-grid

A Board-grid object is a collection of Row-grid objects. Each Row-grid object's horizontal property is true. Adjacent Row-grid objects are positioned one on top of the other.

Properties:

containerId: a unique integer identifying a container of Tile objects.

defStack: the default Tile-stack object

count: the no. of Row-grid objects

rows: a list of Row-grid objects of length equal to the count property

dimensions: width, height (in pixels)

position: left, top (in pixels)

offset: distance in pixels between adjacent Row-grid objects (height = offset multiplied by count)

visible: if false, Board-grid is not visible to user

Animated Game Prototype

Animated Game Classes

The main difference between animated games and board games is that Tile-stack objects are animated and can move about the screen. The User Tile-stack usually contains just one Tile 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

Tile

A Tile object can be a circle, square, or printable character. A printable character can be a letter, digit, or punctuation mark (ASCII 33 - 126). Every Tile 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 Tile object. A Tile object is always contained within a Tile-stack object.

Properties:

value: 0 = empty, 1 = circle, 2 = square, 33 – 126 = printable char. (ASCII value); a negative value indicates Tile 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 Tile object

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

Tile-stack/Vector

A Tile-stack object is a stack of Tile objects. Every Tile object is contained within a Tile-stack object.

A Vector object is a Tile-stack 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 Tile objects. If zero, the Tile-stack object is contained in a Row-grid object.

cards: a list of Tile objects

dimensions: width, height (in pixels)

position: left, top (in pixels)

offsets: leftOffset, topOffset (in pixels); distance between adjacent Tile 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 Tile 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 Tile-stack object

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

canDrag: if false, user cannot drag Tile objects away from this Tile-stack object

canDrop: if false, user cannot drop Tile objects onto this Tile-stack object

infinite: if true, top Tile object is duplicated rather than moved when user drags Tile object away from this Tile-stack object

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

visible: if false, Tile-stack is not visible to user

multiDrag: if true, user can drag more than one Tile object away from this Tile-stack at once. If false, user can only drag away the top Tile object.

isDefault: if true, all properties of this Tile-stack are set equal to the properties of the defStack object of the parent Row-grid object.

Row-grid

A Row-grid object is a row or column of cells, and each cell contains a Tile-stack 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 Tile objects.

defStack: the default Tile-stack object

count: the no. of cells (Tile-stack objects)

cells: a list of Tile-stack objects of length equal to the count property

dimensions: width, height (in pixels)

position: left, top (in pixels)

offset: distance in pixels between adjacent Tile-stack 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, Row-grid is not visible to user

Board-grid

A Board-grid object is a collection of Row-grid objects. Each Row-grid object's horizontal property is true. Adjacent Row-grid objects are positioned one on top of the other.

Properties:

containerId: a unique integer identifying a container of Tile objects.

defStack: the default Tile-stack object

count: the no. of Row-grid objects

rows: a list of Row-grid objects of length equal to the count property

dimensions: width, height (in pixels)

position: left, top (in pixels)

offset: distance in pixels between adjacent Row-grid objects (height = offset multiplied by count)

visible: if false, Board-grid is not visible to user

Navigation:

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

MazeGrid

A MazeGrid object is similar to a Board-grid object, except every cell has 0 to 4 visible borders, or walls (default width = 3 pixels). Most Tile-stacks, including the User Tile-stack, cannot move between adjacent cells separated by a wall. When a Vector other than the User Tile-stack 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 Tile-stack 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 Tile-stack 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 Tile or Tile-stack 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 Tile(s) or Tile-stack
- **OnSubtract** – loses old Tile(s) or Tile-stack
- **OnCollide** – collision detected
- **OnHitWall** – collision with wall detected
- **OnClick**
- **OnDbClick**
- **OnMouseUp**
- **OnMouseDown**
- **OnMouseMove**
- **OnKeyUp**
- **OnKeyDown**
- **OnKeyPress**