

# **TileGamer**

## **for the XO Laptop**

By Mike Hahn

Copyright © 2009 TileGamer.com

Date Last Edited: 19-May-2009

# Table of Contents

<b>Overview .....</b>	<b>1</b>
<u>Introduction .....</u>	<u>1</u>
<u>Background .....</u>	<u>1</u>
<u>Treescript Overview .....</u>	<u>2</u>
<b>Implementation Plan .....</b>	<b>3</b>
<u>Parts of TileGamer .....</u>	<u>3</u>
<u>Help Wanted .....</u>	<u>4</u>
<b>TileGamer Part Descriptions .....</b>	<b>5</b>
<u>Code Editor .....</u>	<u>5</u>
<u>Layout Editor .....</u>	<u>5</u>
<u>Bitmap Editor .....</u>	<u>7</u>
<u>Treescript .....</u>	<u>7</u>
<u>Cross-Compilers .....</u>	<u>8</u>
<u>Game Server .....</u>	<u>8</u>
<u>Game Client .....</u>	<u>8</u>
<u>Non-Programming .....</u>	<u>9</u>

# Overview

## Introduction

[ [Home](#) ]

TileGamer is a tool for developing multiplayer board games and animated games. All games are developed in a built-in scripting language called Treescrypt. The Tilegamer Integrated Development Environment (TIDE) is implemented in Python, enabling it to run on various platforms, such as Windows, Linux, and the XO Laptop. The TIDE is used to develop games. TIDE for the XO includes a cross-compiler which translates Treescrypt into Python. TIDE for Windows and Linux includes a cross-compiler which translates Treescrypt code into a Java applet. The game designer uploads the Java applet code to the TileGamer.com web site, which automatically compiles it into byte-code, enabling end-users to play those games in their web browsers. End-users using their XO Laptops download the compiled Python code to play games. The game server running on the TileGamer.com server can communicate with the Python-based game servers hosted on XO Laptops, enabling XO users to play games with Windows and Linux users, as well as other nearby XO users.

The TIDE enables both programmers and non-programmers to develop multiplayer TileGamer games, and the Treescrypt scripting language is designed to be easier to master than more general-purpose languages like Python and Java. The ultimate goal of TileGamer is to facilitate the emergence of hundreds (or even thousands) of Windows, Linux, and XO Laptop game designers, and thousands more TileGamer end-users. Initially the target platforms will be Linux and the XO Laptop, and later Windows and maybe Macs will be targeted.

## Background

TileGamer is an evolution of an earlier project for the XO Laptop called Treenimation/Boardwalk, which never really made it to the implementation stage (although I learned Python by beginning to implement a small part of that project). I have been working on TileGamer and its predecessors for over 10 years, working sporadically in my spare time. The implementation of TileGamer will begin in the very near future. I have recently scaled back my day job as a Delphi programmer: from now on I will be doing technical writing and maintaining setup scripts, and little or no Delphi programming (after I finish my present project, which involves testing and writing onscreen descriptions of a few dozen reports written in Delphi).

Recently my sister gave me a book called *Three Cups of Tea*, which is about a mountaineer who, grateful to the inhabitants of a remote village who rescue him after he fails to summit K2, promises to build them a school. Eventually he builds over 50 schools in northern Pakistan. That inspired me to resurrect my Treenimation/Boardwalk project. I had previously abandoned my targeting the XO Laptop, instead targeting only Windows (and using Java to implement TileGamer). I was uncomfortable with Python, and I had an old version of Linux which was unreliable. Now I have a more recent version of Linux (Ubuntu). Python isn't so bad, I now realize, and I feel that TileGamer has the potential of bringing software development skills to the masses (namely, schoolchildren in developing countries).

## Treescript Overview

All Treenimation games are written in a built-in scripting language called Treescript. Treescript is a powerful object-oriented language which is designed with beginner programmers in mind. Pressing the question mark (?) key when in the code editor brings up a popup menu of choices valid in the context of the text cursor position. Optional Structure-Editor mode eases code entry for naïve users. The default operator/operand mode is prefix (operators come before their operands) as opposed to optional infix (binary operators come in between their operands). Treescript is based on Java, and when infix mode is selected, Treescript code strongly resembles Java, with a touch of Object Pascal thrown in for good measure.

### Language Features

- **Syntax:** Treescript is a subset of Java, although the syntax differs radically from Java, in that all operators are, by default, prefix (like Lisp) rather than infix, and parentheses are used for grouping.
- **Semicolons:** Statements and declarations are semicolon-delimited.
- **Case:** Treescript is case-sensitive: the convention for identifiers is all lower case, hyphens being used to separate parts of multi-word identifiers. Class names are an exception: the initial letter is upper case.
- **Keyboard Aid:** When enabled, this feature allows the user to enter a hyphen followed by a lower case letter by typing an upper case letter (but only when the text cursor is in the middle or at the end of an identifier). Alternatively, the user may enter a hyphen by typing a quote ('). Also, commas and periods are immediately converted into parentheses if desired (if the period key is used to enter ')', then the ')' key may be used to enter a decimal point). The user may toggle Code Menu mode by typing slash (/) instead of question mark (?). Keyboard Aid is always disabled inside comments and string literals.
- **Meta-Programming:** Treescript programs are lists, which can act as data for other Treescript programs. Structure editor mode lets newbies create simple event handlers without having to know the syntax of the Treescript programming language.

# Implementation Plan

## Parts of TileGamer

The following outline illustrates the various components of the TileGamer system. Each bottom-level component (leaf) can be implemented by one or more Python programmers. If there is more than one programmer, one of those programmers is designated as the lead programmer for that bottom-level component. Each component which is not a leaf (contains other components) is associated with a programmer who is responsible for that component, and who is chosen from the lead programmers of its lower-level components. The same programmer can implement more than one component.

```
Code Editor (
  Structure Mode (prefix)
  Free-Form Mode (
    Prefix
    Infix
  )
  Code Menu Mode
)
Layout Editor (
  Static (
    Tile
    Tile-stack
    Grid (
      Board-grid
      Row-grid
      Table-grid
      Hex-grid
    )
  )
  Animated 1.5-D (
    Tile
    Tile-stack
    Grid (
      Board-grid
      Row-grid
      Table-grid
      Hex-grid
    )
  )
  Animated 2-D (...)
  Animated 2.5-D (...)
  Animated 3-D (...)
  Object Inspector
)
Bitmap Editor
Treescript (
  Treescript Compiler
  TIL Loader
  TIL Runtime
)
```

```
Cross-Compilers (
  Treescrypt to Java
  Treescrypt to Python
)
Game Server (
  Java (web-based)
  Python (XO-based)
)
Game Client (
  Java (web-based)
  Python (XO-based)
)
Non-Programming (
  Web Design
  Web Programming
  Testing
  Writing Documentation
)
```

## Help Wanted

If you wish to help implement TileGamer, and know Python programming (no Python experience needed for final 5 components), please send me an email. Please tell me a little about your Python programming or related experience, including your level of proficiency (beginner, intermediate, advanced, or guru), and select up to 12 of the components (see previous section) that you wish to implement, sorted in descending order of priority (highest priority first). The components you select need not be leaf components (may contain other components). If more than one programmer wishes to implement the same component, I will try to sort out who does what. I will reply to everyone within 30 days. Please send your email to *mike (at) treenimation (dot) net*. Thanks for visiting the TileGamer web site!

# TileGamer Part Descriptions

## Code Editor

The Treescrypt code editor features syntax highlighting and unlimited undo. The user is always in one of three modes: Structure, Free-Form, and Code Menu. Free-Form mode has 2 operator modes: Prefix and Infix.

### Structure Mode

Operator Mode is always Prefix (operators come before their operands). Cursor keys (up, down, left, right) move highlighting to (previous, next, parent, first/current child). Pressing a printable char. key incrementally selects matching menu item (backspace undoes that feature). Pressing Enter/Spacebar selects incremental menu item, if any. Otherwise, display text cursor, insert space after/before cursor. Pressing Ctrl+Enter inserts a newline char.

### Free-Form Mode

Operator Mode is either Prefix (operators come before their operands, the default), or Infix (operators come between their operands). Press Ctrl+O to toggle Operator Mode. Press Esc to toggle Structure/Free-Form Mode.

### Code Menu Mode

Press Question Mark (?) to enter/exit Code Menu Mode. Press Esc to show/hide code menu.

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).

Cursor keys (up, down, left, right) move selection up/down or select parent/child menu (if any). Pressing a printable char. key incrementally selects matching menu item (backspace undoes that feature).

Pressing Enter/Spacebar goes to lower level menu, or if none, insert current menu item and go to next menu item, or if none, go to parent code menu. If Spacebar was pressed originally, exit Code Menu Mode.

## Layout Editor

The Layout Editor is used to lay out tiles (cards or game pieces), tile stacks, grids (each grid cell contains one tile stack), and standard widgets.

### Standard Widgets

Labels, buttons, edit boxes, check boxes, radio buttons, combo boxes, memo boxes, panels, group boxes, image boxes, tabbed notebooks, etc.

## **Static Components**

These components are used for board games, and do not support animation.

### **Tile**

A collection of bitmaps, labels (text), and/or vector graphics. A standard playing card (Card) is a subclass of the Tile class.

### **Tile-stack**

A linear array of (possibly overlapping) Tile objects. Tile-stacks may support multi-select, in which the user may select a particular Tile object in the stack by clicking/dragging on it with the mouse.

### **Grid**

A one- or two-dimensional array of cells. Each cell contains one Tile-stack object.

### **Board-grid**

A two-dimensional array of cells. Each cell contains one Tile-stack object, and all cells are the same size.

### **Row-grid**

A one-dimensional array of cells. Each cell contains one Tile-stack object, and all cells are the same size. This grid may be oriented horizontally or vertically.

### **Table-grid**

A two-dimensional array of cells. Each cell contains one Tile-stack object, column widths and row heights may vary, adjacent cells may be merged into one cell, and any cell may be split vertically or horizontally into 2 or more cells.

### **Hex-grid**

A two-dimensional array of cells. Each cell contains one Tile-stack object, and all cells are the same size and shape. Each cell is shaped like a hexagon or a triangle.

## **Animated Components**

There are 4 classes of animated components: 1.5-D, 2-D, 2.5-D, and 3-D. Tile objects in 1.5-D can only move in 4 directions: up, down, left, or right, and are always either centered in a Tile-stack object or straddling the border between 2 adjacent Tile-stack objects. Tile objects in 2-D can be located anywhere within the borders of a Tile-stack object (or can straddle the border between 2 adjacent Tile-stack objects), and can move in any direction, but vector graphics are not supported. Tile objects in 2.5-D add support for scalable vector graphics, including linear transformations. Tile objects in 3-D are composed of triangles in 3-dimensional space, with a color or texture. The XO Laptop may not be powerful enough to support 3-D Tile objects.

In addition to the usual mouse events, animated components support collision-detection events, which are triggered when 2 Tile objects in the same Tile-stack collide, or when a Tile object collides with a Tile-stack border. Both static and animated games have optional joystick support, which is treated as cursor key events, other key events (Enter, Spacebar), or mouse click events.

## **Tile-stack (animated)**

A collection of (possibly overlapping) Tile objects. Tile-stacks may support multi-select, in which the user may select a particular Tile object in the stack by clicking/dragging on it with the mouse.

## **Grid (animated)**

A one- or two-dimensional array of cells. Each cell contains one Tile-stack object.

## **Object Inspector**

A rectangular window with 2 columns and multiple rows. The 1st column contains a list of properties, and the 2nd column contains a list of values. Each property has its own data type, and the values of different data types are edited in different ways. The cells of the Object Inspector are populated whenever the user selects a game component by clicking on it. If multiple game components are selected, only those properties common to all selected components are displayed. If the Events tab is selected, the first column contains a list of events.

## **Bitmap Editor**

The Bitmap Editor is used to edit bitmaps, which are usually contained within Tile objects or image boxes.

## **Treescript**

Treescript is the name of the TileGamer built-in scripting language. All TileGamer games are based on Treescript code, even simple drag-and-drop games created by non-programmers.

## **Treescript Compiler**

The Treescript Compiler takes as its input the Treescript code entered in the Code Editor and translates it into a lower-level language called Tilegamer Intermediate Language (TIL), which is a text file.

## **TIL Loader**

The TIL Loader takes as its input the TIL code generated by the Treescript Compiler and loads it into memory in the form of a tree. Each node in the tree is composed of 10 bytes: a 2-byte header, and a pair of 4-byte data/address values.

## **TIL Runtime**

The TIL Runtime executes the in-memory tree generated by the TIL Loader, and supports debug settings (single-stepping, stepping over, stepping out of, and breakpoints).

## **Cross-Compilers**

### **Treescript to Java**

Translates Treescript code into a Java applet (source code, not byte-code).

### **Treescript to Python**

Translates Treescript code into Python (source code, not compiled code). Not included (or disabled) in TileGamer distributions targeting platforms other than the XO Laptop.

## **Game Server**

### **Java (web-based)**

Based on Project Darkstar (PDS), which is an advanced game server written in Java. This game server runs on the TileGamer.com server, and keeps all game players in any given game process synchronized. Communicates with the game clients in the form of XML files sent back and forth.

### **Python (XO-based)**

Each XO Laptop user involved in a given game process communicates with each of the other players via the Python game server, which is hosted locally and keeps all game players synchronized. Communicates with the other game servers in the form of XML files sent back and forth using the wireless peer-to-peer networking.

## **Game Client**

### **Java (web-based)**

Hosted locally, the game client sends and receives XML files to and from the game server, which runs on the TileGamer.com server.

### **Python (XO-based)**

Generates and parses the XML files sent to and received from the game server, which is also hosted locally.

# **Non-Programming**

## **Web Design**

The Web Designer is involved with laying out the TileGamer.com web site in an aesthetically pleasing and functionally smooth fashion, enabling guests to kibitz on games in progress, play games in single-player mode, and register (enter full name, user ID, password, email address, and optional security question/answer if they have no email address). Registered users must log on (enter user ID and password) in order to play games and download/install the TIDE. Support for game scheduling (making play dates with other users) is included. Available games are organized alphabetically, by genre, and may be sorted by popularity/no. of games in progress and searched for in various ways.

After the users select a game, they enter a game lobby and join an existing table or start a new table. For games such as chess and bridge with fixed nos. of players, the game starts automatically when the table is full. Otherwise, each player must click on Start before the game can start. The first player is selected randomly or in a game-dependent fashion. Players can chat with each other by clicking on the chat window and entering a message. Many games such as chess support game ratings/rankings, which are measurements of how skillful/lucky each player is rated. Various forums are included, such as technical support, bug reporting, designing games, different game genres, game-specific forums for the more popular games, etc.

## **Web Programming**

The Web Programmer, working closely with the Web Designer, decides what language/database to use for the TileGamer.com web site, and implements all web-based functionality, including the interface between the web-based game server (written in Java) and the rest of the web site.

## **Testing**

The Software Tester is responsible for testing all TileGamer components, including the Code Editor, Layout Editor, Static/Animated Components, Bitmap Editor, Treescript Compiler, TIL Loader/Runtime, Cross-Compilers, and Gamer Server/Client. All bugs found are to be reported to the appropriate Python programmer. Another responsibility of the Software Tester is to make sure the web site is intuitive and flows smoothly.

## **Writing Documentation**

The Technical Writer documents the design specs of all TileGamer components (see above: Testing), writes user manuals for game designers and end-users, converts these manuals into PDF documents, as well as HTML-based online help. Working closely with the Web Designer, the Technical Writer ensures that all labels and blocks of text displayed to visitors of the TileGamer web site are free of typos and English mistakes, and are clear, meaningful, and well laid out.

## **Business Model**

All non-XO Laptop users are given the option of becoming Patrons (paying \$24/year), or remaining Casuals (playing for free). Whenever the user name of a Patron is displayed anywhere, it is accompanied by a yellow star. Whenever the user name of a Casual is displayed, it is accompanied by a green triangle. All XO Laptop users play for free, and their user names are unaccompanied by green triangles.