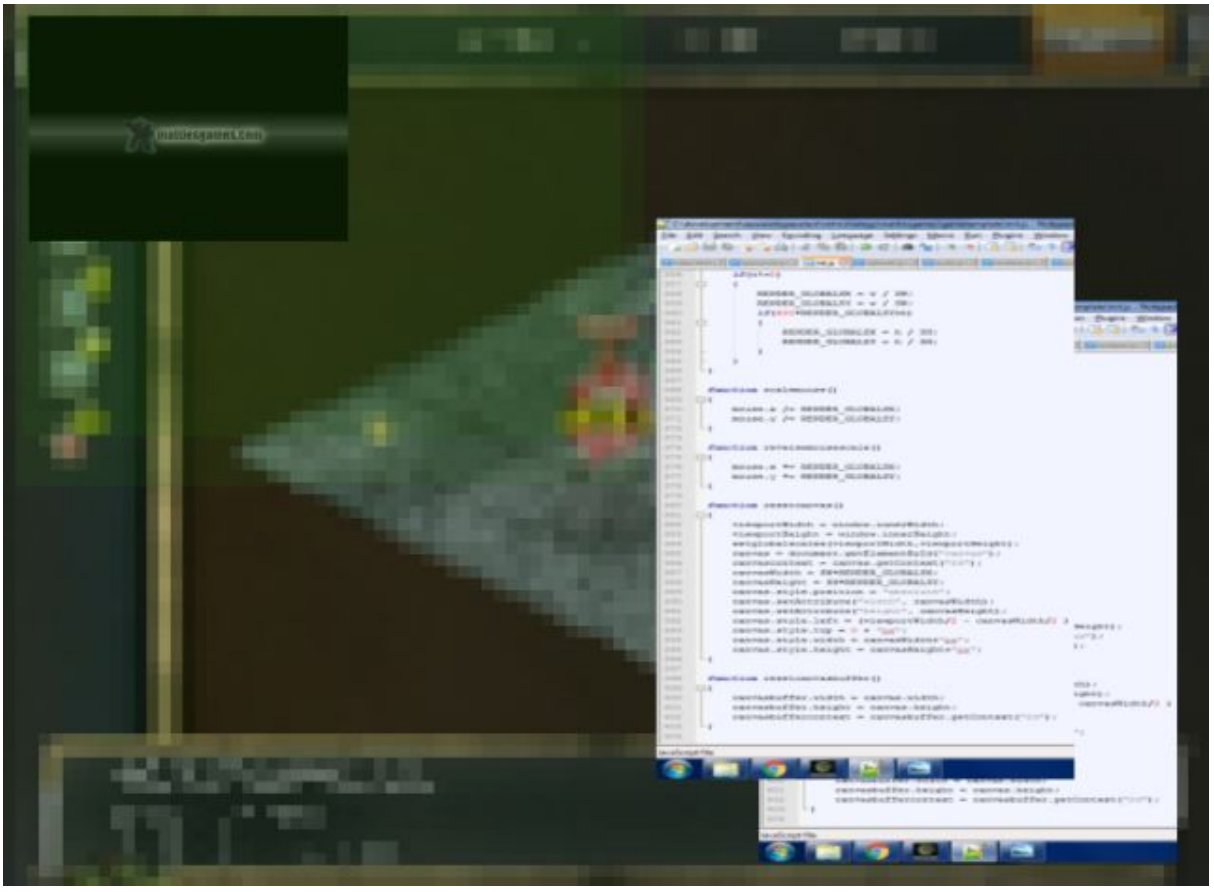
mattiesgames.com - game engine documentation - 2020



Written by Matt Lloyd

Table of Contents

Introduction

Mattiesgames.com Javascript game engine is written purely in javascript.

This document describes the various modules that are found (.js files) in the engine and how they are intended to be used.

The order of the modules is important and will be dealt with in this document in the order in which they are called in the index.html file.

Most of the modules will rarely change, and when an update occurs to say the renderer.js module this should not affect the other modules manner of interacting with them.

For example - if improvements are made to the renderer.js file then simply replacing the old renderer.js with the new one should be compatible in most cases.

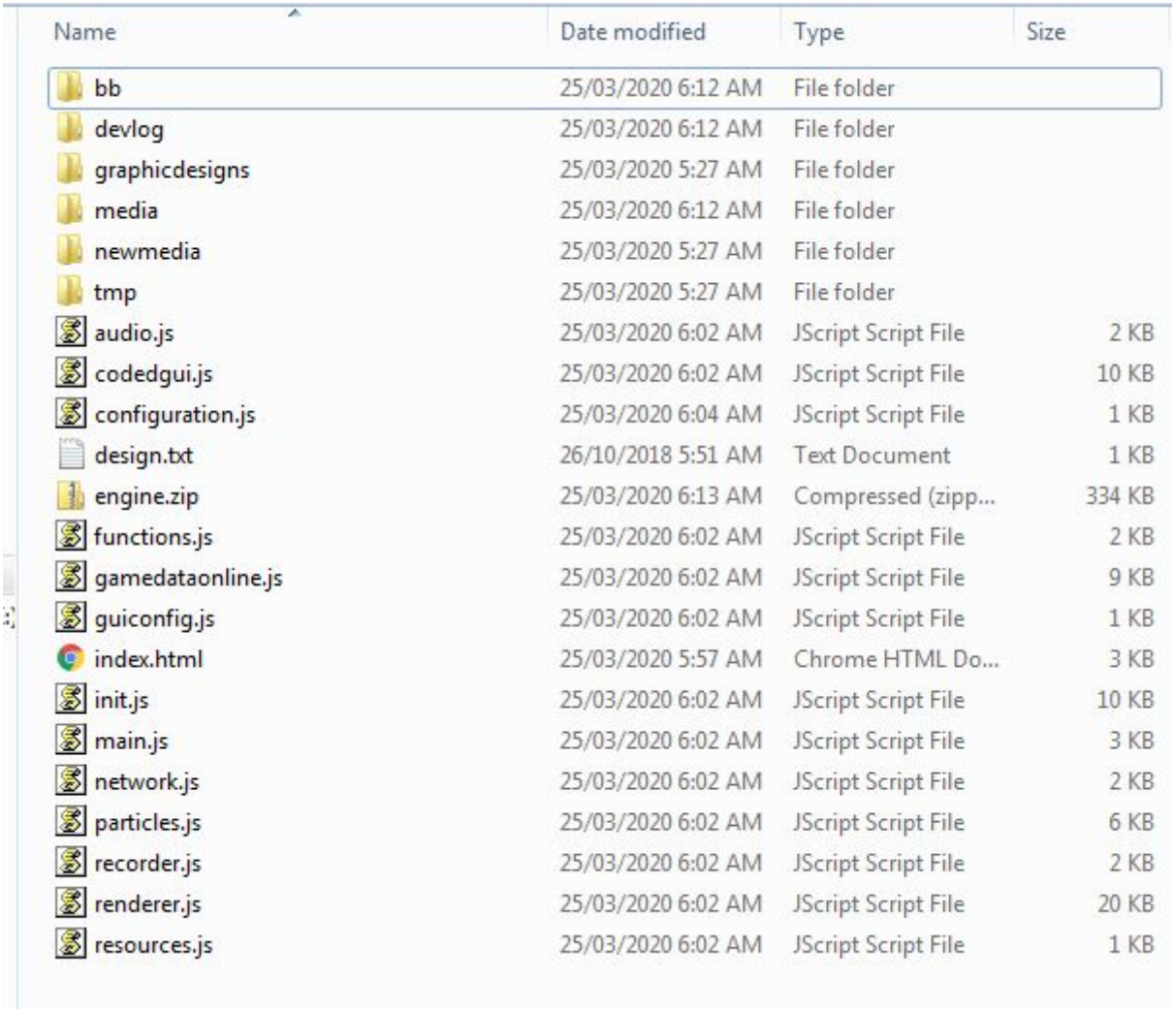
The website http://www.mattiesgames.com/main/ contains the engine in its base form, with minimal media.

Before describing the individual modules, the layout of the folder structure plus its initial contents before beginning development on a game are shown below.

The main folder of your game should look like this. The folders marked ‘bb’, ‘devlog’, ‘graphicdesigns’, ‘newmedia’ and ‘tmp’ are helper folders - they do not strictly necessary.

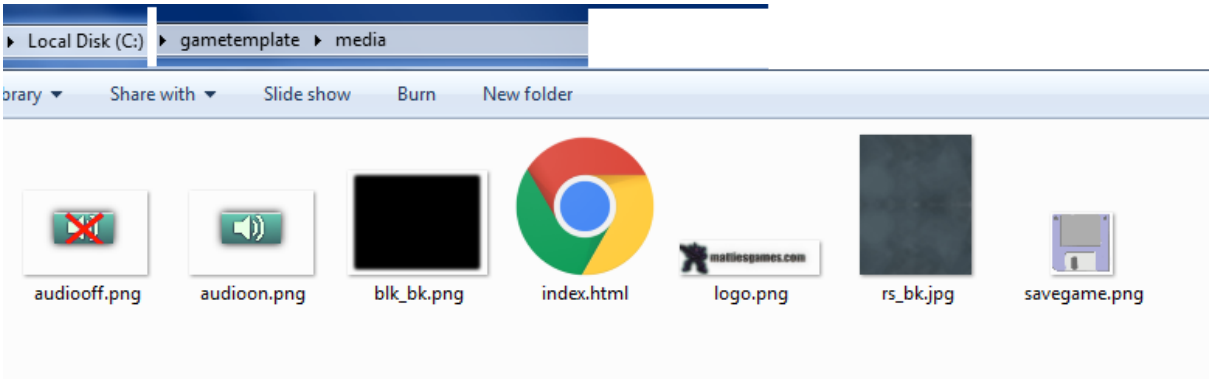
The folder marked ‘media’ is necessary.

The .js and .html files below are required to run the game engine.



Name	Date modified	Type	Size
bb	25/03/2020 6:12 AM	File folder	
devlog	25/03/2020 6:12 AM	File folder	
graphicdesigns	25/03/2020 5:27 AM	File folder	
media	25/03/2020 6:12 AM	File folder	
newmedia	25/03/2020 5:27 AM	File folder	
tmp	25/03/2020 5:27 AM	File folder	
audio.js	25/03/2020 6:02 AM	JScript Script File	2 KB
codedgui.js	25/03/2020 6:02 AM	JScript Script File	10 KB
configuration.js	25/03/2020 6:04 AM	JScript Script File	1 KB
design.txt	26/10/2018 5:51 AM	Text Document	1 KB
engine.zip	25/03/2020 6:13 AM	Compressed (zipp...	334 KB
functions.js	25/03/2020 6:02 AM	JScript Script File	2 KB
gamedataonline.js	25/03/2020 6:02 AM	JScript Script File	9 KB
guiconfig.js	25/03/2020 6:02 AM	JScript Script File	1 KB
index.html	25/03/2020 5:57 AM	Chrome HTML Do...	3 KB
init.js	25/03/2020 6:02 AM	JScript Script File	10 KB
main.js	25/03/2020 6:02 AM	JScript Script File	3 KB
network.js	25/03/2020 6:02 AM	JScript Script File	2 KB
particles.js	25/03/2020 6:02 AM	JScript Script File	6 KB
recorder.js	25/03/2020 6:02 AM	JScript Script File	2 KB
renderer.js	25/03/2020 6:02 AM	JScript Script File	20 KB
resources.js	25/03/2020 6:02 AM	JScript Script File	1 KB

The contents of the media folder on a new game being developed are shown as below.



Now that we know the folder structure and contents we can begin working our way through each of the modules and describing their function and usage.

Running and Installing

Simply place the folders and files as explained above and run the ‘index.html’ file. On doing so the game engine will begin. You should see the logo appear with a little animation followed by a black screen.

It will look something like this:

Configuration.js

This file contains a handful of important constants for the game itself.

They can be changed during development and their usage is fairly self explanatory.

var DEBUGGING = true; //change this to false when no longer debugging.
var PGAME = 'mygame'; //change this for each game. //this is your game name for the purpose of metrics reporting.
var METRICSACTIVE = false; //this determines whether or not metrics are being reported.
var FRAMERATE = 60; //the framerate of the game in frames per second.

Resources.js

This file contains a single function call ‘loadall()’

Inside this function you ought to place all resources to be loaded, either audio or visual using the following functions to do so:

loadmusic()
loadaudio()
loadimage()
loadanimimage()

The arguments for each of these functions are described here:

loadmusic(index,filename);

Index is a sequential number for all audio files. You need to begin at zero and work your way up. It is shared between all audio files, so the same sequence needs to be used for both loadmusic and loadaudio.

The filename is a standard .ogg, or .mp3 or other audio file.

loadaudio(index,filename)

As per loadmusic - same rules apply. However the difference internally between loadmusic and loadaudio is that music is set to loop (although this can be changed), and it has a flag marking it internally as a music clip as opposed to a sound clip.

loadimage(index,filename)

Same rules as per music - except that the files to be used are either .png or .jpg files.
loadanimimage(index,filename,cellwidth,cellheight,cellswide,cellshigh);

Index and filename are the same as for the other resource loading functions.

Cellwidth and cellheight are the individual frames of the animation strip/atlas that are being used.

Cellswide and cellshigh are the number of rows/columns in the animation strip.

For example;

loadanimimage(1,'media/spacemonster.png',48,64,8,6);

Would load an image of a space monster, with a total of 48 frames - 8 across and 6 down, with each frame being 48x64 pixels in size.

Init.js

This file contains most of the processes for setting up the canvas, the input devices and other features for the game.

It is also where the resources are loaded for the game.

The main things you will wish to change are these lines shown below:

```
var VERSIONINT = 1;
var AUTHOR = 'Your Name Here';
var YEAR = 'Year of Development';
var SW = 800;
var SH = 600;
var FULLSCREEN = true; //change this and the above SW/SH will be used as screen resolution....
```

The values are fairly self explanatory.

Some of the functions you will use quite often are:
mousex(), mousey() and leftclicked() which are the positions of the mouse x, y coordinates and whether or not the mouse has been clicked.

There are also functions for leftdown() and leftup() to indicate the status of the mouse at a particular point in time.

Recorder.js

This is currently not used. It is an attempt at generating playback/replay/record features using mouse input commands per frame. It is not documented because it is not currently in a state that is suitable for usage.

Network.js

This module handles the processes that fire on game launch and during the game to record various metrics for reporting. It can be switched off by setting the METRICSACTIVE flag to false in the configuration.js.

The main metrics that are sent are:

Launch - when the game fires.
TimeStamp - when the game has been in progress once per minute while active.

Custom events can be sent to the server with sendmetric(event,value) which sends a datapoint for a specific event (string) and a specific value (string). Usage of this is for your own purposes.

Audio.js

This module handles the processing of audio in the game. The main function calls that are of benefit to developers are:

playmusic(index)
playaudio(index) or playsound(index) (identical function, either name may be called)

The index refers to the index found in the loadmusic and loadaudio function calls described earlier.

The volume of the audio is set with variable MUSICVOLUME.

Other functions available are:

stopallaudio() which halts all audio playing.

loopaudio(index) and unloopaudio(index) which sets up a sound to be looped or not. It does not actually play the sound.

Renderer.js

This module handles all screen drawing. You can draw lines, unfilled and filled rectangles, images, text and animated images.

The basic method of drawing is within the main drawgame() function (gamedataonline.js) you would call the various drawing operations in the order you wish them to appear on the screen. In the main.js file the screen is then rendered and flipped into view each frame.

Within the module the useable function calls are documented but I will repeat that here:

drawline(sx,sy,fx,fy,color,blend)
Draws line or plots point with color ('#RRGGBB') and blend mode (blend to be discussed further down)

drawtext(x,y,text,color,font,centre)
Draws text at x,y, with color '#RRGGBB' and font eg '20px Arial' at centre true/false

drawunfilledrect(x,y,w,h,color,blend)
Draws unfilled rect with colour at x,y and w,h and blend mode

drawrect(x,y,w,h,color,blend)
Draws filled rect with colour at x,y, and w,h and blend mode

drawanimimage(img,x,y,angle,scalex,scaley,framenumber,blend,istext,alpha,index)
Draws an animated image frame with img at x,y with angle (radians), and scalex/y, using framenumber (0-n), with blend, istext should be false, alpha 0-1, and index = images[] array index number

drawimage(img,x,y,angle,scalex,scaley,framenumber,blend,istext,alpha,celltype)
As for drawanimimage except framenumber is ignored, celltype is ignored

renderworld()
Call once per frame at end of all drawing to issue all draw commands to canvas

flip()
Call once per frame to flip the canvas into view

cls(color)
Clear the screen with colour as shown

getcolor(r,g,b)
Return a colour value as a string of 3 pairs of hexidecimal numbers eg '#FFFFFF' would be getcolor(255,255,255).

Blend modes in the renderer.js are either 0 (no blending) or 1 (additive / brightening).

Angles sent to the drawimage commands for rotations are in radians. To convert from degrees to radians, multiply your angle in degrees by Pi and divide by 180. This will give a value for a full circle being 2*Pi, which is the same as turning through 360 degrees.

Guiconfig.js

This module contains some basic configuration for user interface elements. Colours, fonts and so on are detailed here.

Codedgui.js

This module contains and handles the user interface interactions in the game. The main functions involve button pressing, text labels, and panels.

Button pressing is handled by calling one of either two functions - drawbuttongrad() or drawbutton().

In each case arguments are supplied indicating the starting upper left x/y position and the finishing lower right x/y position as well as the title of the button, if any, and also in the case of drawbuttongrad() a set of arrays of colours that determine the appearance of the button.

They return true on the button being clicked, and false otherwise.

They also have a hover mode where they draw a highlighted version of themselves.

Functions.js

This module contains some basic common functions including a random number generator with a known seed that can be set, some simple mathematical functions, and conversions between screen space and world space for isometric games.

The benefit of using the random number generated here, rather than your browser's built in generator is that by setting the seed you can be sure of getting the same result on each pc that runs this code. Useful if you wish to procedurally generate levels, worlds and so on the same each time by setting a simple seed value.

Gamedataonline.js

This module is where most of the action takes place.

In each game you develop this is where you will place your constants, your functions and your variables and arrays and so forth that handle your game logic.

There are a number of functions that already exist in this module, that make life easier for you.The main function that everything else hangs off is ‘drawgame()’.
Also the game relies on a series of ‘gamestates’. A variable called ‘gamestate’ holds an integer value referring to the current game state - be it a menu screen, a gameplay screen or otherwise. By switching between gamestates you can control game processing within the drawgame() function call.

A typical part of the drawgame() section of the code will look like this:

```
if(gamestate==GAMESTATE_MY_MODE)
{
	//draw your images
	//detect input with mouse commands (leftclicked(), mousex() and mousey() )
	//do logic…..
}
```

In terms of timing the game has a current ‘frame’ value stored in ‘frame’ which can be used for timing events and such. Your FRAMERATE setting will determine how many frames elapse before one second passes.

Main.js

This is the final module to be called in the index.html file. It contains the final preparatory steps to run the game and set up the update loop.

There are no functions in here that you need to call.

Appendix 1

Here are some images from some games that were created using this game engine.


Warlock
Monsters:25
Day:0
Gold:65
Advisor
WHACK
Skeleton Warrior rolls a 12 against DC:19 to HIT.Fail!
Phasga Oskvet rolls a 24 against DC:14 to HIT.Success!
Phasga Oskvet earns 24xp, 3 gold pieces.
Skeleton Warrior dies.
Skeleton Warrior rolls a 9 against DC:19 to HIT.Fail!
Pause
+
Character


HI SCORE
27
SCORE
27
WIZ ARENA
2


ADVISOR
PASS
ORDER
Nadia the great assassin has sided with you in the battle against Chaos, her blade will be most effective.