

## Contents

---

Contents .....	1
Version History .....	3
Introduction .....	3
Important concepts .....	3
Setting up a game project.....	3
Source code.....	3
Primer .....	3
tictactoe.html.....	4
tictactoe.js .....	4
Files .....	5
Module: <b>Engine</b> .....	6
Class: TinyGameEngine .....	6
Properties.....	6
Methods .....	7
Module: <b>Flipbook</b> .....	9
Class: Sequence.....	9
Properties.....	9
Methods .....	10
Module: <b>Audio</b> .....	11
Class: Sounds.....	11
Properties.....	11
Methods .....	11
Class: SFX.....	13
Properties.....	13
Methods .....	13



## Version history

---

Number	Date	Notes
1.0	7.7.2021	Initial (Added: Setup, Files, Class:TinyGameEngine)
1.1	8.7.2021	Added: Class:Flipbook
1.2	12.7.2021	Improved layout, added Primer code, Class:Audio
1.3	???	ADD: Actor class

## Introduction

---

Tiny Game Engine or TGE for short is a lightweight and flexible game engine targeted for modern HTML5 browsers. The code is written in 100% JavaScript using Await/Async model and Promises to handle asynchronous execution (as opposed to traditional callbacks).

While the engine is not limited to a certain genre, it is written primarily with beginner/intermediate programmers in mind and most of the functions attempt to only cover relatively simple 2D retro style arcade action games. TGE does not implement true 3D rendering, utilize WebGL API nor provide any server-side functionality.

For advanced 3D and/or multiplayer games I would recommend using frameworks like Three.js and Babylon.js. These frameworks are orders of magnitude larger systems and require good programming skills, deep understanding of 3D maths and rendering, GLSL shading language, physics engines, etc.

### Important concepts

Many concepts, class, and method names in TGE are derived from Epic Games' Unreal Engine.

The main class of TGE is the [Engine](#) class, which is instantiated automatically by default. It acts as a container for all other subsystems. Two other important concepts are 1) [GameLoop](#) class, which also automatically created as a property of [Engine](#). [GameLoop](#) in turn contains all the game world entities, referred as 2) [Actor](#)(s). There are several useful descendant classes of [Actor](#), such as [Player](#), [Projectile](#) and [Enemy](#).

## Setting up a game project

---

### Source code

<https://playpointgames.com/engine/v1/>

### Primer

A simple barebones of a game project follows. While it does not do anything interesting, it represents a good starting point for a game project. Several basic game systems are introduced, such as: main game loop, audio subsystem and keyboard controls. It also demonstrates the creation of a basic [Player Actor](#).

The code execution starts from the bottom of the [primer.js](#) code when the page has been fully loaded and DOMContentLoaded event fires. First, Engine's main HTMLElement is set and [InitGame\(\)](#) is called.

Once the [Player](#) object has been created and [Audio](#) subsystem is initialized, the game loop is started. [Engine.start\(\)](#) method accepts one optional parameter, an event handler which is executed every "tick" of the main gameloop. The tick rate defaults to 60 times per second. *This is the place where you should execute all your game logic.*

## primer.html

```
<!DOCTYPE html>
<html>
  <head>
    <title>Tic-Tac-Toe</title>
    <meta charset="UTF-8">
    <link rel="preconnect" href="https://fonts.gstatic.com">
    <script src="https://playpointgames.com/js/ae2.js"></script>
    <script src="https://playpointgames.com/api/ppgapi.js?v=1.1.0"></script>
    <link rel="stylesheet" type="text/css" href="css/tictactoe.css">
    <link href="https://fonts.googleapis.com/css2?family=Ranchers&display=swap"
rel="stylesheet">
  </head>
  <body>
    <div id="game">

    </div>
    <script type="module" src="primer.js"></script>
  </body>
</html>
```

## primer.js

```
import * as TGE from '/engine/v1/engine.js';
import { Player } from '/engine/v1/player.js';
import * as Utils from '/engine/v1/utlis.js';
import * as Audio from '/engine/v1/audio.js';

const { Engine, Actor, Types } = TGE;    // create shorthand names to required classes
const Vec2 = Types.Vector2;              // ...and make reference to Vector2 even shorter

/*
  Main game loop
*/
function tick() {
  const keys = player.controllers['keyboard'].keyState;

  if (keys.up)    player.position.add(new Vec2(0, -1));
  if (keys.right) player.position.add(new Vec2(1, 0));
  if (keys.left)  player.position.add(new Vec2(-1, 0));
  if (keys.down)  player.position.add(new Vec2(0, 1));
}

/*
  Set up the player actor and initialize the game systems
*/
async function initGame() {
  // create player
  const player = Engine.gameLoop.add('player');
  player.create({ container:ID('game'), className:'player' });
  player.movement.acceleration = 0;
  player.attachKeyboard();

  // place the player in the middle of the viewport
  player.position.set({ x:Engine.screen.width/2, y:Engine.screen.height/2 });

  // wait for mouse click (otherwise audio playback is not allowed by the browser)
  await Utils.waitClick(document.body);

  // init audio subsystem
  new Audio.Sounds(Engine);

  // start the game loop
  Engine.start(tick);
}

/*
  Run this code when the page is fully loaded
*/
AE.addEvent(window, 'DOMContentLoaded', async _ => {
  Engine.setRootElement('game');
  await initGame();
});
```

## Files

Name	Modified	Classes	Description
engine.js	4.7.2021	TinyGameEngine	Engine (the main class of TGE)
actor.js	23.6.2021	Actor	Actors are living entities in the game. Examples: character, enemy or player.
childActor.js	23.6.2021	ChildActor	Part of an actor.
animations.js	29.11.2020	Animations	HTML5 Video API encapsulation.
root.js	16.2.2021	Root	Ultimate ancestor object for Actor
canvasRenderer.js	27.6.2021	CanvasRenderer	Subsystem for rendering graphics on Canvas (instead of using HTML elements).
flipbook.js	7.7.2021	Sequence, Flipbook	Frame by frame animation.
utils.js	9.4.2021	-	Utility methods.
world.js	4.7.2021	TileMap, Scene, World	Subsystem for tile-based game worlds.
multicast.js	24.5.2021	-	Maintains event stacks.
gameLoop.js	24.5.2021	GameLoop	Main gameLoop which updates the game logic (Tick event) and graphics (Update event)
gameController.js	11.3.2021	KeyController, GamepadController, PointerController, AllGamepads, AllGamepadControllers, Controllers	Encapsulates different types of game controllers: Keyboard, Gamepad, Pointer.
debug.js	28.2.2021	-	Creates a debug information/performance analysis layer (HTMLElement)
particles.js	2.4.2021	ParticleSystem, Emitter	Subsystem for spawning large amount of copies of a single image and describe how they evolve through time.
mainmenu.js	26.5.2021	MainMenu, MenuItem	Main menu subsystem.
audio.js	12.4.2021	Sounds, SFX	Encapsulates Web Audio API.
colliders.js	11.2.2021	Collider	Collision subsystem for Actor classes.
player.js	17.2.2021	Player	a descendant of Actor class.
projectile.js	6.1.2021	Projectile	a descendant of Actor class.
texture.js	7.7.2021	Texture	a class which encapsulates an Image and provides access to its pixels via OffscreenCanvas (falls back to Canvas)
parallax.js	11.1.2021	Parallax, Layer	Collection of overlaid HTMLElements which can be scrolled at different speeds creating an illusion of depth.
physics.js	11.2.2021	Enum_PhysicsShape, PhysicsShape, Circle, AABB, Box, Poly	Subsystem for managing collision responses of 2D geometries.

## Module: Engine

---

File: engine.js

### Exports

Name	Type	Desc
Engine	Class	
World	Class	
Scene	Class	
Actor	Class	
Collider	Class	
Root	Class	
Enum_HitTestMode	Enum	
Enum_ActorTypes	Enum	
Types	Module	
Utils	Module	

### Class: TinyGameEngine

Creates and initializes two singleton classes: Engine (main class of the TGE) and GameLoop (container for game actors) objects. Installs default event handlers for basic keyboard and mouse interactions.

### Properties

Name	Access	Value	Description
<a href="#">styleSheet</a>	R	CSSStyleSheet	Returns TGE's internal CSS stylesheet.
<a href="#">isMobile</a>	R	Boolean	Returns TRUE if browser window has PORTRAIT-PRIMARY flag set.
<a href="#">zoom</a>	R/W	Number	Gets and sets the current browser Zoom level.
<a href="#">mousePos</a>	R	Vector2	Returns the current pointer position.
<a href="#">LMB</a>	R	Boolean	State of the left (primary) mouse button.
<a href="#">RMB</a>	R	Boolean	State of the right (secondary) mouse button.
<a href="#">version</a>	R	String	Current version of the engine.
<a href="#">hasWorld</a>	R/W	Boolean	Has the World object been created for this engine instance? Setting the flag true will create one.
<a href="#">hasEdges</a>	R/W	Boolean	Should the Actors respect the boundaries set by the engine.edges rectangle?
<a href="#">hasContextMenu</a>	R/W	Boolean	Enables and disables the browser context menu (disabled by default)
<a href="#">aspectRatio</a>	R	Number	Ratio of screen width / height

## Methods


getFPS()		
Return current FPS rate.		
Return Value	Type	Desc
Anonymous	Number	Returns average FPS over past 30 frames.

setMouseCallbacks(o)		
Installs user defined mouse callbacks.		
Param. Name	Type	Desc
o.mousedown	Function	
o.mouseup	Function	
o.mousemove	Function	

setKeyCallbacks(o)			
Installs user defined keyboard callbacks.			
Param. Name		Type	Desc
o.keydown		Function	
o.keyup		Function	
o.keypress		Function	

setFullscreen(value)		
If parameter <i>value</i> equals <b>true</b> , this method sets the <b>engine._rootElem</b> as full screen target. If the engine is already set in full screen using this method, and <i>value</i> is set to <b>false</b> , it will exit full screen.		
Param. Name	Type	Desc
value	Boolean	Sets fullscreen mode true or false.

setFlags(o)		
Copies flags defined in parameter object <b>o</b> into the Engine flags register object.		
Param. Name	Type	Desc
o.hasWorld	Boolean	
o.hasEdges	Boolean	
o.hasAutoAdjustScreen	Boolean	
o.preserveAspectRatio	Boolean	
o.hasContextMenu	Boolean	
o.mouseEnabled	Boolean	

 start(beforeRenderCallback)	
Starts the GameLoop. Optional callback function may be supplied, which will be called prior to processing of each frame.	

GameLoop updates physics (if enabled), updates the Actors and responds to Controller input.

#### `pause()`

Pauses the GameLoop. Frames are not rendered, and physics are not updated while in pause mode.

Player Controllers will remain enabled and respond to events. Otherwise, the user would not be able to exit pause mode using his/her controller.

If this behavior is not desired, Controllers must be detached or deactivated.

#### `resume()`

Resumes the execution of GameLoop. Callback defined by previous call to `engine.start()` will be respected.

#### `_onResizeWindow(e)`

Private event handler. Called internally.

#### `_onContextMenu(e)`

Private event handler. Called internally.

`updateFlags(o)`

`recalculateScreen()`

`setRootElement(el)`

`addEvent(evtName, callback)`

`addSVG(parentElem, tagName)`

`createWorld(o)`

`addScene(o)`



## Module: Flipbook

---

File: flipbook.js

Exports

Name	Type	Desc
Sequence	Class	Single Animation sequence.
Flipbook	Class	Container for image files and animation sequences attached to an Actor.

### Class: Sequence

#### Properties

Name	Access	Value	Description
<code>loop</code>	R/W	Boolean/Number	Looping status and/or loop iteration count for the sequence.
<code>FPS</code>	R/W	Number	Frames per second override for the sequence (overrides owner Flipbook FPS setting). The given value will be clamped between 0 and 60.
<code>frame</code>	R	Number	Current frame as Integer.
<code>length</code>	R	Number	Frame count of the sequence.
<code>index</code>	R/W	Number	Current frame as Double (contains fractional part). Set by the constructor.
<code>start</code>	R/W	Number	Starting frame of the sequence. Set by the constructor.
<code>end</code>	R/W	Number	Ending frame of the sequence. Set by the constructor.
<code>direction</code>	R/W	String	<p>Playback customization. One of the following values are supported: <code>forward</code>, <code>backward</code>, <code>forward-reverse</code>, <code>backward-reverse</code>.</p> <p><code>forward</code> plays the sequence forwards from <code>start</code> frame to <code>end</code> frame and increases internal iteration count. This is the default.</p> <p><code>backward</code> plays the sequence from <code>end</code> frame to <code>start</code> frame and increases the iteration count.</p> <p><code>forward-reverse</code> same as <code>forward</code> but when the last frame is reached, the animation is played in reverse, returning back to starting frame before increasing the iteration count.</p> <p><code>backward-reverse</code> same as <code>backward</code> but when the first frame is reached, the animation is played in normal direction, returning back to last frame before increasing the iteration count.</p> <p>The value is not managed. Anything can be written in it. The system recognizes keywords "forward" and "reverse" when interpreting the content of the string.</p>

## Methods

### `play()`

Starts playback of the sequence. Returns a promise which resolves immediately if `loop` value is set to `True` or `Infinity`.

### `stop()`

Stops playback of the sequence.

### `seek(frame)`

Seeks to a frame by setting `index` equal to `frame` parameter. The parameter must be a Number or the method will silently fail. Before overwriting `index` the value is clamped between `start` and `end` frames.

### `clone()`

Returns a clone (copy) of the current sequence.

### `next()`

Do not call manually. This method is used by the owner `Flipbook` to advance the internal `index` of the animation sequence.

### `_nextIteration()`

Private method. Called internally.

### `resetCycle()`

Resets the animation cycle to starting condition. Called internally by `play()` method. There should be no need to call this manually.

## Module: Audio

---

File: audio.js

Exports

Name	Type	Desc
Sounds	Class	Audio subsystem object (singleton).
SFX	Class	Encapsulation of a single audio file.


### Class: Sounds

Audio subsystem.

### Properties

Name	Access	Value	Description
N/A			



### Methods

 <b>constructor(engine)</b>		
Creates the audio subsystem and adds a reference to it in <a href="#">Engine.audio</a> property.		



Param. Name	Type	Desc
engine	TinyGameEngine	Reference to <a href="#">Engine</a> object.

 <b>play(o)</b>		
Starts playback of a track (audio file).		

Param. Name	Type	Desc
o.name	String	Unique name for the track (user managed).
o.volume	Number	Normalized (0..1)
o.delay	Number	In seconds.
o.startTime	Number	Number of seconds to skip from the start of the file.
o.loop	Number	Integer: number loops to play.

  <b>add(name, url)</b>		
Adds a new track (audio file) into the internal collection.		

Param. Name	Type	Desc
name	String	Unique name for the track (user managed).
url	String	URL of the audio file.

  <b>addBunch(list)</b>		
Adds a new track (audio file) into the internal collection.		

Param. Name	Type	Desc
list	Array of object	List of anonymous objects which consists of two fields: name and url. See method <a href="#">add(name, url)</a> .

<a href="#">mute()</a>
Mutes all Audio files created using the audio subsystem.

<a href="#">clear()</a>
Deletes all tracks from the internal collection and frees up the resources.

<a href="#">seek(name, seconds)</a>		
Sets the playhead position in an audio file.		
Param. Name	Type	Desc
name	String	Unique name for the track (user managed).
seconds	Number	Position to move the playhead to, in seconds, from start of the audio file.

<code>fadeOut(time)</code>		
Fades out all tracks in the internal collection and stops them when their volume reaches 0. This is a convenient way to fade out all audio – for example – when a level ends.		
Param. Name	Type	Desc
time	Number	Fadeout time to complete silence (in milliseconds).


## Class: SFX

Wrapper for HTML Audio element. Represents a single audio file referred as *track*. The object can be upgraded to use Web Audio API using [upgrade\(\)](#) method.

## Properties

Name	Access	Value	Description
<a href="#">volume</a>	R/W	Number	Gain value of the track. Examples: 0 = Muted 1 = Full volume
<a href="#">currentTime</a>	R	Number	Current position of the playhead in seconds.
<a href="#">pan</a>	R/W	Number	Panning of the track. Examples: -1 = Full Left 0 = Middle (Default) 1 = Full Right Note: Requires a prior call to <a href="#">upgrade()</a> .
<a href="#">rate</a>	W	Number	Set playback rate of the track. Examples: 0.5 = Half speed 1.0 = Full speed (Default) 2.0 = Double speed

## Methods



 <a href="#">constructor(sounds, o)</a>		
Creates a new track in the <a href="#">sounds</a> object's internal collection.		
Param. Name	Type	Desc
sounds	Sounds	Reference to <a href="#">Sounds</a> object which owns this SFX.
o.file	String	URL of audio file.
o.name	String	User defined, unique name for the track.
o.onLoaded	Function	Optional. Callback function executed when the audio file has completed loading.


 <a href="#">play(loops)</a>		
Starts playback of the audio file.		
Param. Name	Type	Desc
loops	Number	Optional. Integer. Number of times the track should be played.

 <a href="#">stop()</a>		
Stops playback of the audio file.		

<a href="#">mute()</a>		
Mutes the audio file. Playback continues normally.		

<a href="#">destroy()</a>		
Destroys the object and releases memory.		

  <a href="#">upgrade()</a>
Upgrades the object to utilize Web Audio interface which enables panning and finer control over audio effects.

 <a href="#">fade(o)</a>		
Linearly slides audio volume up or down.		
Param. Name	Type	Desc
o.duration	Number	Duration of the effect in milliseconds.
o.targetVolume	Number	Normalized target volume level. See <a href="#">volume</a> property.
o.resolution	Number	Optional. How often the volume is updated, in milliseconds (should be less than duration). Setting this to a larger number requires more frequent callbacks. Typically, 50 milliseconds (Default) is enough for a smooth volume slide.

## Module: Actor

---

File: actor.js

Exports

Name	Type	Desc
Actor	Class	Ancestor class for all game world entities.
Enum_ActorTypes	Enum	Enumeration of different (built-in) Actor types.

### Class: Actor

Actors are an essential part of most game types. They represent entities in the game world which have their individual graphics, transform (position, rotation and scale), movement and several other properties.


### Inheritance

Actor → Root

### Properties

Name	Access	Value	Description
N/A			

### Methods

 constructor(o)		
Creates an instance of Actor. The parameter object <b>o</b> is optional.		
Param. Name	Type	Desc
o.owner	GameLoop	GameLoop which owns this Actor. Typically, Actors are created using <a href="#">GameLoop.add()</a> method which fills in this parameter automatically.
o.position	Vector2	Initial position of the Actor.
o.rotation	Number	Initial rotation angle of the Actor (in degrees).
o.scale	Number	Initial scale of the Actor.
o.onClick	Function	Function to call when Actor is clicked with pointer device (mouse).
o.onTick	Function	Function to call on every game logic update frame. Use sparingly since this might have a big performance impact.
o.onBeginOverlap	Function	Function to call when this Actor overlap with another Actor begins.
o.onEndOverlap	Function	Function to call when this Actor overlap with another Actor ends.
o.data	Object	Property values of <a href="#">o.data</a> will be copied into the Actors internal <a href="#">data</a> field. The internal <a href="#">data</a> field is for user data and TGE will never access the information.
o.name	String	User defined name for the Actor. This can be used for searching Actors from <a href="#">GameLoop</a> containers. No checking is done, multiple Actors may share the same name.

addChild(o)		
Add a new <a href="#">ChildActor</a> into this Actor's <a href="#">children</a> collection. The parameter object <a href="#">o</a> is optional.		
Param. Name	Type	Desc
o.position	Vector2	Initial position of the Actor.
o.rotation	Number	Initial rotation angle of the Actor (in degrees).
o.scale	Number	Initial scale of the Actor.
o.elemType	String	Name of the HTML element created for the ChildActor.
o.name	String	User defined name for the <a href="#">ChildActor</a> . This can be used for searching ChildActors from the parent Actor's <a href="#">children</a> collection. No checking is done, multiple ChildActors may share the same name.

getChildByName(name)		
Returns the first ChildActor from this Actor's children collection whose <a href="#">name</a> property matches the <a href="#">name</a> parameter.		
Param. Name	Type	Desc
name	String	Name of the <a href="#">ChildActor</a> to search for.
Return Value	Type	Desc
Anonymous	<a href="#">ChildActor</a> or <a href="#">null</a>	If a <a href="#">ChildActor</a> is found, returns a reference to it, otherwise returns <a href="#">null</a> .