Game Scripting, Being in Control of your Game

The TiledGame.exe is a simple C# based game running a single level of GamePlay.

It contains mechanics for loading a Tiled style tmx level and facilitates firing triggers, playing sounds and running timers. But the mechanics do not make the game. It needs Game Logic to make it into a playable game. The GameLogic is handled by the lua script. Lua is a simple and fast scripting language that is used widely in games to enable designers and players to modify games.

So it's the Game Designers job to bring life to the game. All he needs is the TiledGame executable, assets and tools like Tiled to help him designing the Level.

The lua script is called game.lua and is to be found in the "assets" directory. You can change this as neede, save it and run the TiledGame.exe.

To Design the level Tiled is preferred. The TiledGame needs a few things :

- TileSets to provide the needed graphics.
  A tile that an Objects can move over should have the property Walkable set to true.
  A tile having the property trigger will cause the function with the provided name called in your lua script.
- One Layer to specify the tiles of the level. Note the level can be very big, so you could create several areas in the single level. The behavior of your tiles depend on the Properties you've setup in the tileset.
- Objects define the free moving objects. If you provide them the Property Trigger, they cause the trigger function to be called in the lua script when hitting something
- ImageLayers are used for Hud purposes. The TiledGame handles the "Score", "Health" and "Message" imagelayer and in the lua script you can write score, health and messages centered in these areas. Remember to name your imagelayers correctly.

It is advices to have one Layer, one Objects and three ImageLayers in your tmx. Save your tmx using Tile Layer Format being CSV.

Lua : withing Lua you can use the general lua language constructs and the common libraries. Additional there is an TiledGame api, a set of functions to manipulate the game.

TiledGame Lua API :

The TiledGame script has about 25 functions available for the game designers needs. Some have return values others do not. The syntax gives the return value first (if any) and then Library.Function and between () the needed extra information, the arguments.

Camera.Follow(Object folowee)

The Camera will try to keep the provided Object in the center of the sreen.

Bool success Controller.Chase(Object chaser, Object chasee, Number speed)

Has the chaser head straight for the chasee with given speed
Returns true when hitting something, otherwise false.

Bool success Controller.Evade(Object evader, Object evadee, Number speed)

Has the chaser head straight away from the evadee with given speed. It's the opposite of chase.
Returns true when hitting something, otherwise false.

Bool success Controller.Move(Object object, Number x, Number y)

Moves the object x, y away from its current position.
Returns true when hitting something, otherwise false.

Bool success Controller.Patrol (Object patroller, Number speed)

Controller has the given object heading straight forward until the move fails, then chooses a new direction randomly.
Returns true when hitting something, otherwise false.

Bool success Controller.Wasd (Object player, Number speed)

Controller sets the controll of the provoded Object to the WASD keys using provided speed.
Returns true when hitting something, otherwise false.

Debug.Log(String message)

Prints the provided message string to the console.

Hud.Health(Number health)

Displays the Health number on screen.

Hud.Message(String message, Number duration)

Displays the Message string on Screen for the Duration number of seconds.

Hud.Score(Number score)

Displays the Score number on screen.

Bool pressed Input.GetKey(Number key)

Checks whether the provided key number is pressed.
Returns true if the key is pressed, otherwise false.
Space = 32, Escape = 27 (Ascii table)

Object firee Level.Fire (Object parent, string name, Number id, String trigger)

Adds a new Object to the level having it's position and direction from the parent with name string, id number from the TileSet.

Object spawnee Level.Spawn (String name, Number id, Number x, Number y, String trigger)

Adds a new Object to the level with name string, id number in the TileSet, the x and y numbers for the position and trigger string for what trigger it fires on collision. If trigger is nil, then no trigger.

Object found Level.GetObject (String name)

Finds the Object with the provided name. If there are more Objects with the same name, the first found is returned.
Returns an Object reference of the first found Object having name, or nil when none found

Table objects Level.GetObjects (String name)

Finds all objects with this name
Returns an array with all found objects, array is empty if non found

Bool inrange Level.InRange(Object observer, Object observee, Number range)

Takes the distance between the center of the two objects
Returns true is the distance is smaller than the given range.

Level.Load(String name)

Loads the level file (tmx) with provided name in the assets directory as the Current level
Returns the ref to the level object (can’t use it), nil on error.

Level.RemoveObject(Object object)

Removes and destroys the Object from the level.
Note, using this Objects ref after this will cause errors!

Bool success Object.Move(Object removee, Number x, Number y)

Moves the object with the x and y step if possible.
Returns true on succes and false if it hits anything

String name Object.GetName(Object object)

Gets the name for the object.
Returns the name string of the object and nil if does not have one.

Number x, Number y Object.GetPosition(Object object)

Gets the current position for the object.
Returns x, y numbers as the current position

Bool success Object.Forward(Object object, Number step) :

Moves the object step number in its current direction

Returns true on success and false if it hits anything

Bool success Object.Property(Object object, String property) :

Moves the object step number in its current direction

Returns true on success and false if it hits anything

Sound sound **Sound.Load(String name, Bool looping, Bool streaming)**

Loads as sound from file with name string and specifies looping boolean it it should be repeating and streaming boolean. Streaming is advised with sounds lasting more then a few seconds, and are not frequently used.
Returns a Sound reference

SoundChannel channel **Sound.Play(Sound sound)**

Plays the Sound
Returns the soundchannel ref which resembles the currently playing sound

Bool isplaying **Sound.IsPlaying(SoundChannel channel)**

Checks if the soundchannel is stil playing or is finished
Returns true if playing, false if finished. Note, a repeating sound will never be finished playing unless you stop the channel.

Sound.Stop(SoundChannel channel)

Stops the soundchannel playing.

Timer timer **Timer.Start(String name, Number interval, Bool repeat)**

Starts a timer with name string as the timeout trigger with interval number and repeat Boolean
Returns the timer ref

Timer.Stop(Timer timer)

Stops the timer from running, it will not produce a timeout trigger anymore

Some lua programming hints

Use comments to clarify your script

```
Level.Remove( enemy ) -- the object will be removed
```

Always start loading a level with Level.Load(“filename.tmx”). Note that the file is looked for in the “assets/” directory relative to the TiledGame.exe

Store player data in an associated array, do the same for other game data. Avoid having loads of unorganized data.

```
local player {
   ref = nil,
   score = 0,
}
```

Use the **Start** function to do initial stuff after the level has been loaded like grabbing the player reference with

```
function Start()
   player.ref = Level.GetObject( “playername” )
end
```

Use Update function for per frame actions like moving objects

```
function Update()
   Controller.WASD( player, 3 )
   Controller.Patrol( enemy, 4 )

   if score == 100 or health <= 0 then -- en when max score or health zero
      Hud.Message( “You’re DEAD\nGame Over”, 5 )
   end
end
```

Provide trigger functions, meaning functions with the name of the trigger, to catch triggers.

```
function BulletHits( target, source )
   if target == player then
      player.health = player.health – 1
   end
   Level.Remove( source ) -- remove the bullet from the level
end
```

Common errors

Using **non existent variables** of functions giving nil error.

In array **forget comma's**

```
player = {
   ref = nil,
   score = 10,
}
```

To be continued….