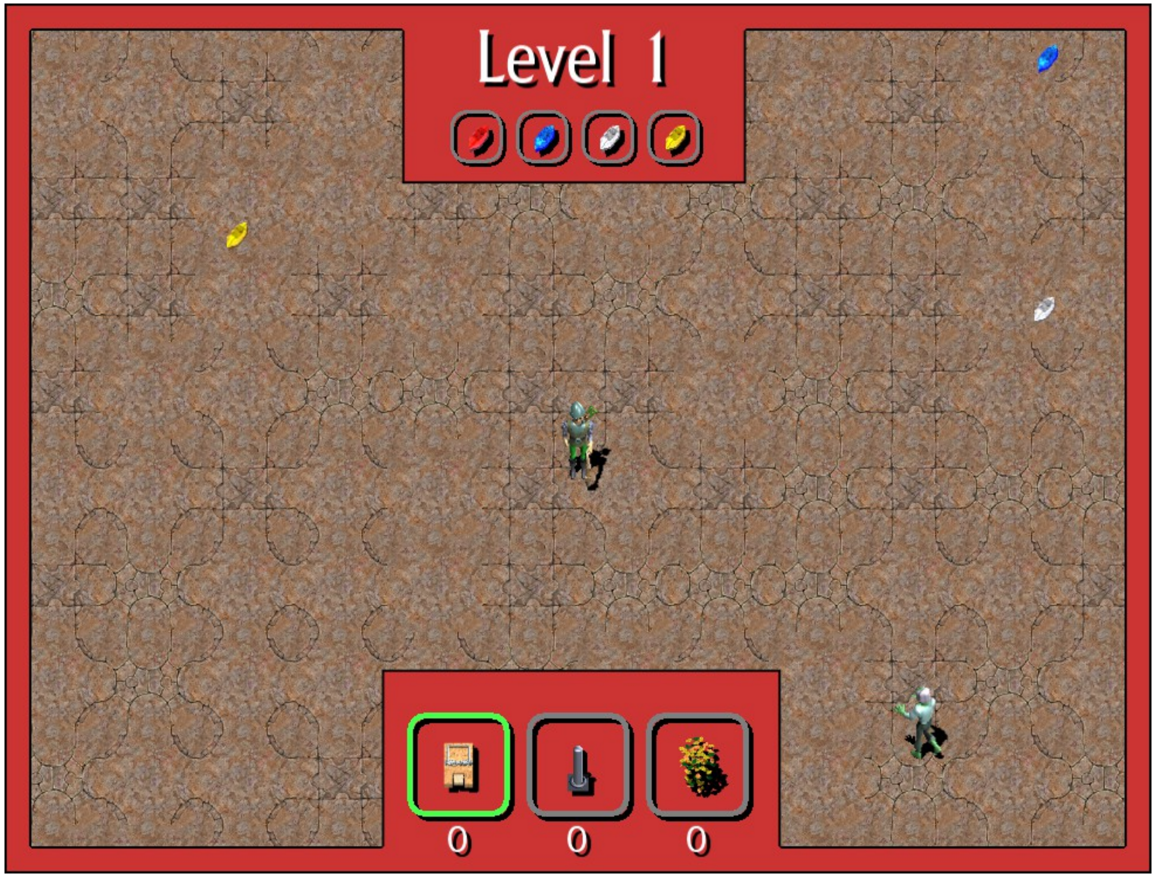
Top Down Shooter (1)



Moving On

This is a quick guide to the first (of two) desktop games to be made for the Corona Geek hangout. While it is called **Top Down Shooter**, it evolved into something a little different from my original intentions. Now that it has gotten to the point where it is playable, I have decided it is time to move on to game two. This second game will be a true top down shooter.

This guide is a sort of 'table of contents' to the game code. I wrote it to help folks locate and identify the major parts of this game.

I suggest you take at least a quick look at this guide and the game because you'll be seeing much of it again in our second game.

Major Game Parts

- Desktop Windowing
- (Rendering) Layers
- Sound
- Player
- Reticle
- Camera
- Enemy Manager
- Sprites
- Terrain
- Game Pad Input
- Keyboard and Mouse (spread through many modules as 'key' and 'mouse' event listeners).

Desktop Windowing

- **File(s):** windowing.lua
- **Purpose:** This code listens for specific 'key' inputs and handles desktop relates actions such as:
  - Switching view mode to full-screen, windowed, maximized, or normal (restor last windowed mode size).
  - Showing Logger Window – A piece of code I wrote so we could see log output while running in desktop mode.
- **Additional Notes:** This code has no function excepted when publishing to OS X or Windows dekstop.

(Rendering) Layers

- **File(s):** layersMaker.lua
- **Purpose:** This code creates (and destroys) a set of rendering layers to keep visual elements correctly separated.
- **Additional Notes:**
  - The layers created by this code are used for camera control.

Sound

- **File(s):** soundMgr.lua
- **Purpose:** This code plays sound effects and a sound track.
  - **Sound Effects** – These are played in response to the global event 'onSFX'.
  - **Sound Track** – The code to play a sound track is called directly from the module.
- **Additional Notes:** We did not talk about this code because we used similar code in prior games. Watch prior game development episodes to learn how to use it.

Player

- **File(s):** playerMaker.lua
- **Purpose:** This code creates the player object. It has these parts:
  - **create()** method –
    - (78..79) Creates player object using the 'greenArcher' sprite maker.
    - (82..89) Initializes flags used below.
    - (92) Starts playing the default animation for player sprite.
    - (94 .. 268) Sets up either keyboard and mouse listeners or controler listeners.
      - These listeners contribute to movement, weapong firing, and dropping of traps.

Player

- **create()** method continued –
  - (274..342) **enterFrame()** listener -This the guts of the player and does the following:
    - (286..279) Selects facing angle based on reticle position.
    - (293..327) Selects proper facing walk / pause / shooting animation.
    - (329..339) – Limits player movement to specific world region.
  - (344..487) **fireArrow()** player method – Generates arrows on demand and moves them in the correct direction.

Player

- **destroy()** method – Stops listening for events set up by create(). Used to clean up before destroying the player.

Reticle

- **File(s):** reticleMaker.lua
- **Purpose:** This code creates a reticle that is used to indicate the direction a player will face, move, and fire arrows. This code handles two input types:
  - Mouse
  - Joystick (re-directed axis events)

Camera

- **File(s):** cameraMgr.lua
- **Purpose:** This code handles keeping the player in the center of the screen. It implements a very simple fixed viewpoint camera. The technique it uses is as follows:
  1) Track initial position of player.
  2) Detect player has moved.
  3) Move group containing player and world in the opposite direction.

This has the effect of making it look like the world moves around the player while it stays in the middle of the screen.

Enemy Manager

- **File(s):** enemyManager.lua
- **Purpose:** This code is responsible for generating new enemies as old enemies are destroyed.
  - It generates more enemies as the game 'level' increases.
  - It uses the green/red zombie makers and the bowskel maker to create sprites. Then attaches 'ai' code to these enemies to make the move and in the case of bowSkels to fire at the player.
  - It 'drop' chests about 20% of the time a enemy is destroyed.
- **Additional Notes:** This code supplies helper functions for use by the leafTrap: getRandom() and getNearest(). These are used to select an enemy to target next.

Sprites

- **File(s):** spriteMaker.lua, arrowMaker.lua, bowSkelMaker.lua, chestMaker.lua, gemMaker.lua, greenArcherMaker.lua, greenZombieMaker.lua, leafStorm.lua, mouseTrapMaker.lua, redZombieMaker.lua, and spikeTrapMaker.lua.
- **Purpose:** These 'makers' create specific sprites used for nearly every visual element in the game. Most of these makers user the 'spriteMaker' module to do most of the work and the supplement that code with code specific to each maker.
  - Ex: mouseTrapMaker.lua uses spritMaker to build the sprite, but adds its own code to control animating it and triggering it when a zombie or bowArcher touches it.

Terrain

- **File(s):** forestMaker.lua, groundMaker.lua, lostGarden.lua.
- **Purpose:** These pieces of code are used to draw the random ground tiles and 'forest' around the play space.
  - **lostGarden.lua** – A module that allows you to draw specific tiles from the (massive) Lost Garden tile set and further more to align them seamlessly.
  - groundMaker.lua – A small module that uses the lost garden module to draw a grid of ground tiles.
  - forestMaker.lua – A small module based on groundMaker that draws a random selection of trees outside the bounds of the play area.

Game Pad Input

- **File(s):** gamePad.lua
- **Purpose:** This code listens for the axis event and then generates a custom onLeftJoystick or onRightJoystick event.
  - This was done because axis events only provide one axis of a single joystick at a time and such information is not useful on its own.
  - The jostick events are cleaned up axis events containing both x/y values for a specific joystick.