SPACE GAUNTLET

Documentation

ver. 0.1.1

04.08.2022

Krikor Arakelyan

1. System requirements

- Python - 3.6.x:
- Pygame - 2.1.x
- Screeninfo – 0.8
- Compatible IDE

2. Installation

- 2.1 Python
  https://www.python.org/downloads/

- 2.2 Pygame
  python -m pip install -U pygame --user

- 2.3 Space Gauntlet
  Unzip the files in a directory of your choice

3. Game Overview

Space Gauntlet is a 2D endless vertical scrolling arcade space themed shooter. The game puts the player in control of a starfighter with the aim of killing as many enemies as they can, before running out of lives.

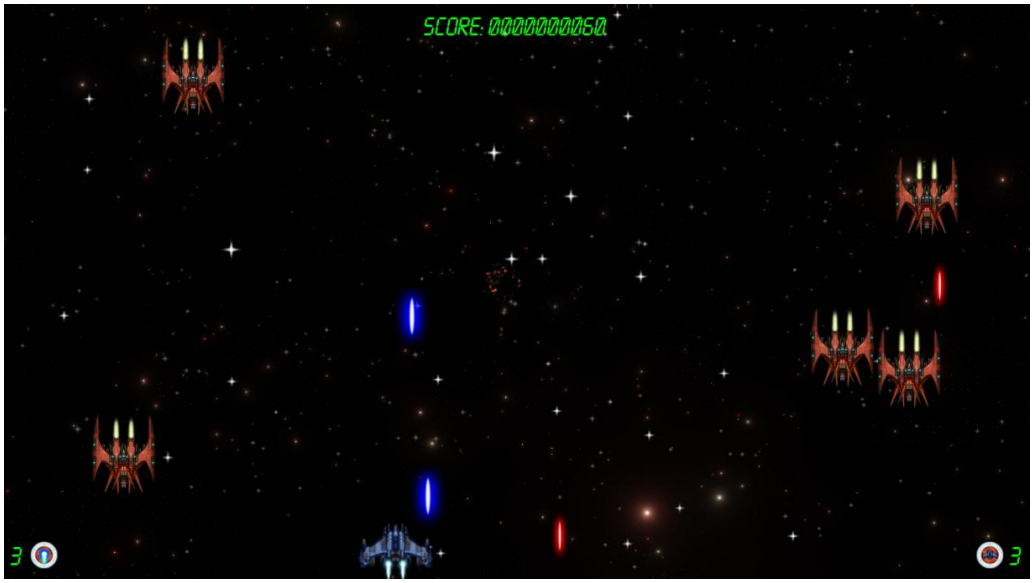
3.1 Menus

Upon running the program, Space Gauntlet's Main Menu will appear.

All navigation in the game's menus happens by pressing the corresponding keyboard button as the mouse pointer is hidden. In every menu, there is a highlighted letter or is written the name of the key to do something or get the player back to the previous screen.

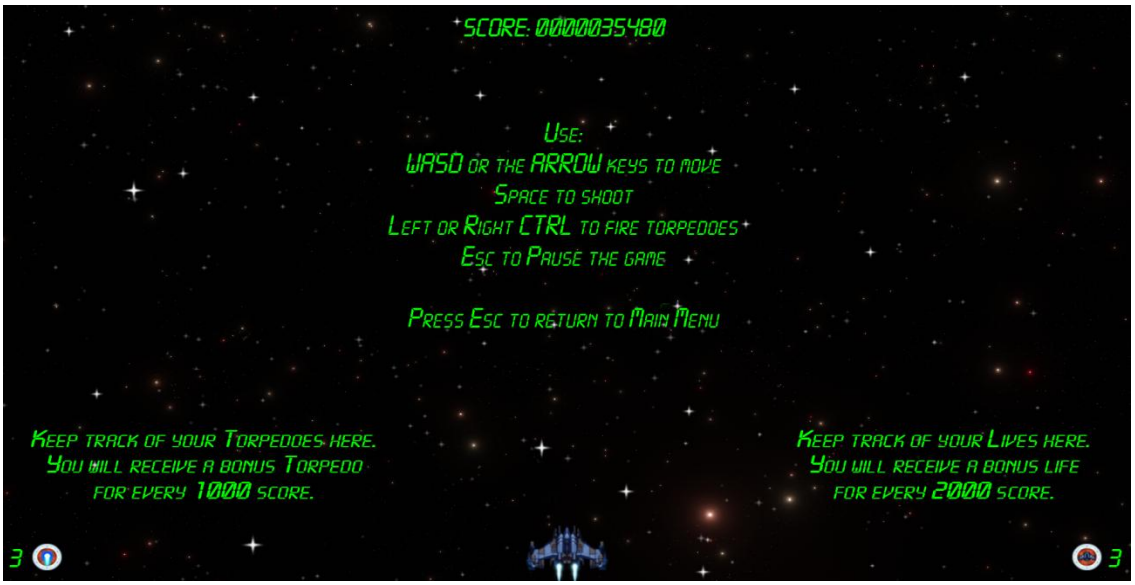
3.1.1 New game

Pressing N starts a new game. The music will change; the player's starship will flicker. When it flickers, the player is invincible. This only happens when starting a New game or on respawn.



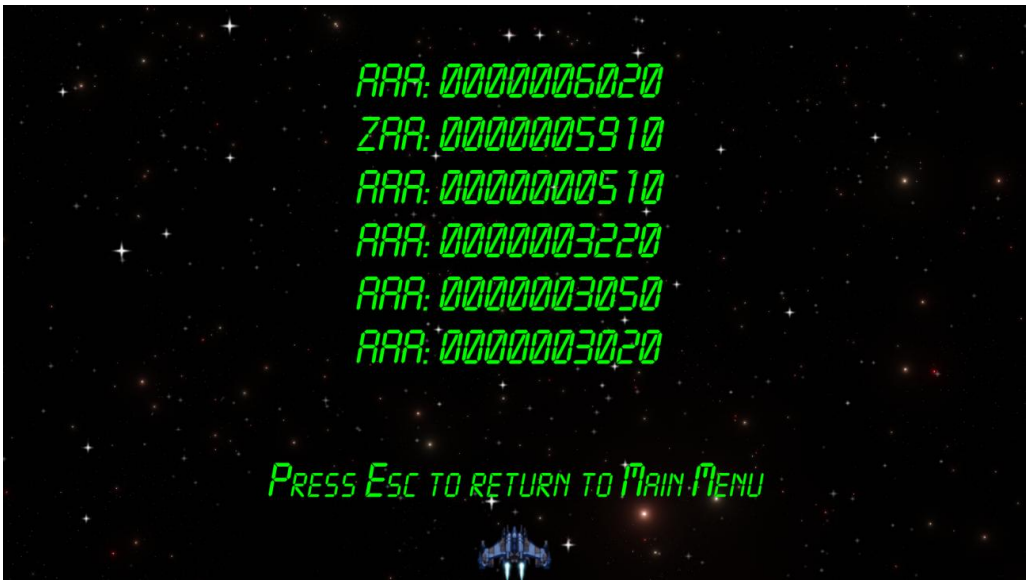
3.1.2 Instructions

A brief explanation on how to play the game.



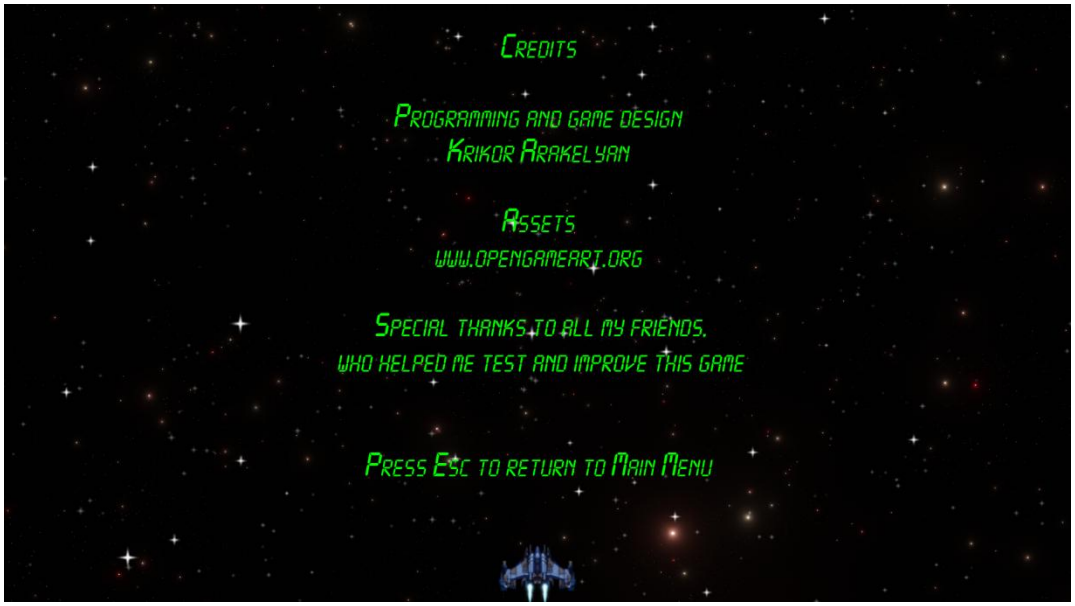
3.1.3 Highscores

The Highscores screen shows the seven best scores. Upon playing the game, the player's score will always appear here, even if they did not beat any of the present highscores.



3.1.4 Credits

This screen shows my thanks to everyone, who contributed to the making of this game.



3.1.5 Quit

Quit exits the game directly, without prompt.

3.1.6 Music and Sound options

Pressing M or S will mute/unmute Music and Sounds. This setting is persistent throughout the Main Menu and New game.

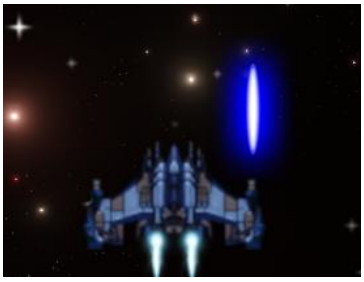
3.2 Controls

3.2.1 Movement

Moving around is done with either WASD or the ARROW keys on the keyboard. The player movement is constrained within the visible area of the screen. Only the enemy ships can leave it. Hitting an enemy ship or being hit by an enemy projectile destroys the player's ship and costs a life.

3.2.2 Primary weapon

Pressing Space fires the primary weapon. The ship alternates fire between the left and right laser guns. Each keypress fires one laser bolt and there is a time delay between shots. Pressing and holding space will still only fire one shot.

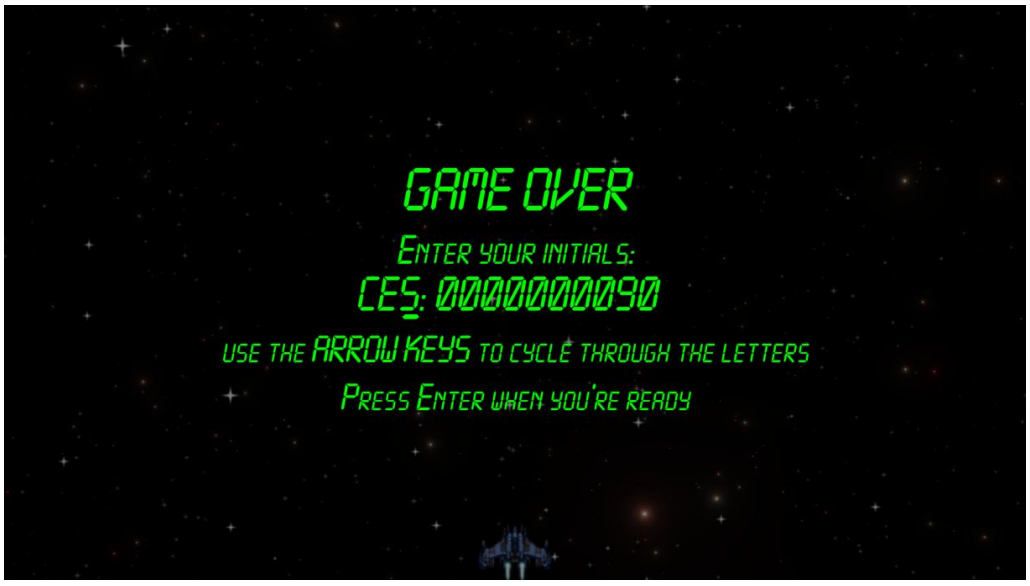


3.2.3 Secondary weapon

The second weapon option are the torpedoes. Their number is indicated on the lower left side of the screen with a Torpedo icon and a number. Torpedoes are launched with either left or right CTRL and if they hit another ship, they destroy all active enemy ships, except the Boss.

3.2.4 "Game Over" screen

When the player runs out of lives, the Game Over screen appears. Here the player can enter their initials using the ARROW keys to cycle through the letters. Upon completion, pressing Enter will bring up the Highscores menu screen. Pressing ESC returns to the Main Menu.



3.3 Difficulty and rewards

3.3.1 Difficulty

Score and number of active enemies regulate the difficulty in the game. The higher the score, the higher the number of active enemy ships at the same time. Explanation on how to regulate the difficulty is in 4.5.1.8new_game.

3.3.2 Rewards and restrictions

The player has three torpedoes per life and receives one additional torpedo for every 1000 score. Upon dying, torpedoes earned by score do not carry over and are lost. The player also gains an extra life for every 2000 score.

3.4 Enemies

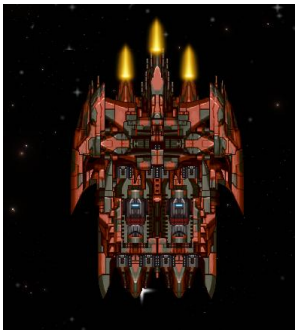
All enemy ships fire at random intervals. Basic enemies spawn at random intervals. All of these characteristics can be changed, as explained in section 4.5.1.8 new_game.

3.4.1 Basic enemy

Alien ship - basic enemy, fires a single laser bolt, moves in a zigzag pattern from the top to the bottom of the screen. Overwhelming in large numbers. Spawns above the visible area of the screen. Destroyed from a single laser bolt or torpedo. Worth 10 points.

3.4.2 Advanced enemy

Advanced alien ship - larger ship, fires two laser bolts at a time, moves from left to right in the top section of the screen. Spawns above the visible area of the screen for every 10 basic enemy ship kills. Destroyed from a 5 laser hits or a torpedo. Worth 100 points.



3.4.3 Boss

Boss - big enemy. Fires 2 or 4 laser bolts. Moves from left to right in the top section of the screen. On a timed move, dives down, then goes back up and continues left-right movement. The Boss takes 20 damage from a torpedo and one from a laser bolt. Spawns above the visible area of the screen for every 3000 score. Worth 1000 score.

4. Code overview.

The code is divided between several files.

4.1 animations.py

Classes and functions responsible for screen animations are in this file.

4.1.1 load_sprite_animations

A function that takes all files from the current directory and its subdirectories and returns a dictionary, where each key is the name of a subdirectory and the values - the names of the files in it.

4.1.2 MovingBackground

A class for generating the moving background animation. Takes the following arguments:

- game_window - screen, on which the background will be drawn
- img_back - background image, displaying slow moving objects that are far from the player
- img_front - foreground image, displaying faster moving objects, that are closer to the player
- speed_back_img - movement speed of back image
- speed_front_img - movement speed of front image
- delta_time - delta time variable for smoother animation

4.1.2.1 update method

This method moves the images, loading two of each, one above the other, thus simulating infinite movement. Can be adapted for horizontal or vertical movement, in both directions.

4.1.2.2 render method

This method draws each frame on the game_window.

4.1.3 Explosion

A class for generating explosions. Takes the following arguments:

- x - x coordinates of destroyed object
- y - y coordinates of destroyed object
- image_list - list of images for the animation

4.1.3.1 update

A method that updates the explosion animation. You can increase the speed of the animation by setting the local variable "image_duration_frames" to a higher integer.

4.2enemies.py

Enemy classes and projectiles are in this file.

4.2.1 EnemyShip

Base enemy class. Takes the following arguments:

- width - Width of screen
- image_list - a list of images for the enemy. There could be only one element in the list, if the enemy is a static image, or multiple, if there is some type of animation for the enemy (e.g. Animated engines on a spaceship)

4.2.1.1 update

A method that updates the animation of the enemy. You can set the speed of the animation by modifying the local variable "engine_animation_speed".

4.2.1.2 move

A method that updates the ship's position, based on its movement pattern. Takes the following arguments:

- delta_time - delta time variable for smoother animation
- height - height of the screen

Enemy movement is created by changing enemy ships x and y coordinates, based on number of moves made or time since last direction change. If an enemy moves below the screen, they respawn above the screen.

4.2.2.3 create_projectile

A method that creates a new projectile class each time the enemy fires their weapons. Takes the following arguments:

- image_laser - an image for the fired projectile
- delta_time - delta time variable for smoother animation
- height - height of the screen

4.2.3 EnemyProjectile

Base projectile class. Takes the following arguments:

- x - x coordinates of projectile origin point
- y - y coordinates of projectile origin point
- image_laser - an image for the fired projectile
- delta_time - delta time variable for smoother animation
- height - height of the screen

4.2.3.1 update

A method that updates the position of the projectile. Enemy projectiles move downwards in a straight line after being fired.

Every projectile that moves below the screen is destroyed to free memory.

EnemyShipAdvanced and EnemyBoss inherit from EnemyShip. EnemyAdvancedProjectile and EnemyBossProjectile inherit EnemyProjectile.

4.3 misc_funcs.py

This file contains various functions for the game.

4.3.1 exit_game

Function to properly close pygame and the game window.

4.3.2 load_sounds

A function that takes all files from the current directory and its subdirectories and returns a dictionary, where each key is the name of a subdirectory and the values - the names of the files in it. Responsible for properly loading the sound files in the game.

4.3.3 load_images

A function that takes all files from the current directory and its subdirectories and returns a dictionary, where each key is the name of a subdirectory and the values - the names of the files in it. Responsible for properly loading the image files in the game.

4.3.4 letter_change

A function that changes the current letter shown on the screen, based on its ASCII value.

4.3.5 letter_pos

A function that changes the letter pointer position based on key press and the letter pointer's current position.

4.3.6 save_new_score

A function that saves the player's score after they finish the game.

4.3.7 high_scores

A function that returns a list of the current highscores

4.3.8 screen_update

A function to update screen, scale and pygame clock. Takes the following arguments:

- window - the main screen of the game
- width - width of the screen
- height - height of the screen
- resolution - resolution variable or monitor resolution tuple(x, y)
- clock - pygame.time.Clock() function

- fps - target fps for the game

4.3.9 mute_unmute_music

A function to mute/unmute background music based on a boolean. Takes the following arguments:

- play - boolean if music is muted or not
- music - audio file
- volume - desired volume if music is not muted

4.3.10 mute_unmute_visualize

A function to visualize if music or sound is muted or not. Takes the following arguments:

- play - boolean if music/sound is muted or not
- screen - game screen to show the image on
- img_mute - image for muted music/sound
- img_unmute - image for unmuted music/sound
- x - x coordinates to display image on
- y - y coordinates to display image on

4.3.11 play_track

A function to change the music track currently playing with a new one. Takes the following arguments:

- music_track - audio file to play
- volume - desired volume for the audio file
- play - boolean if music is muted or not

4.3.12 UserScore

A class to the score with a unique id, to allow for scores with same users' names. Takes the following arguments:

- id_num - unique integer number
- user - initials of the user
- score - score of the user

4.4 player.py

4.4.1 PlayerShip

Main player class, responsible for the player's ship, movement and firing controls. Takes the following arguments:

- width - width of screen
- height - height of screen
- image_list - a list of images for the player’s ship. There could be only one element in the list, if the ship is a static image, or multiple, if there is some type of animation for the ship(e.g. Animated engines on a spaceship)

- image_list_invul - a list of images for the player's ship while invulnerable.There could be only one element in the list, if the effect is a static image, or multiple, if there is some type of animation(e.g. flickering)

4.4.1.1 update

A method that updates the animation of the player's ship. You can set the speed of the animation by modifying the local variable "engine_animation_speed".

4.4.1.2move

A method that updates the ship's position, based on keypress. Takes the following arguments:

- delta_time - delta time variable for smoother animation
- window_limit - size of the screen to constrain the ship within, usually the resolution of the monitor

Movement is constrained within the visible part of the screen with the self.rect.clamp_ip(window_limit) method

4.4.1.3create_projectile

A method that creates a new projectile class each time the player fires their weapons. Takes the following arguments:

- w_type - string with the type of weapon being fired. "laser" or "torpedo"
- image_type - an image for the fired projectile
- delta_time - delta time variable for smoother animation

4.4.2 PlayerProjectile

A class, responsible for the player ship's projectiles. Takes the following arguments:

- x - x coordinates of player's ship
- y - y coordinates of player's ship
- w_type - string with the type of weapon being fired. "laser" or "torpedo". If the w_type is "laser", fire alternates between left and right ship laser guns by using and modifying the class variable fire_left.
- image_type - an image for the fired projectile
- delta_time - delta time variable for smoother animation

4.4.2.1 update

A method that updates the position of the projectile. Player projectiles move upwards in a straight line after being fired. Every projectile that moves above the screen is destroyed to free memory.

4.5 space.py

This is the main Space Gauntlet file. All variables are defined here.

4.5.1 Game

Main class of the game.

4.5.1.1 enemy_shoot_and_move

A method that updates enemy movement and shooting. Takes the following arguments:

- time - time of method call
- enemy_group - group of enemy ships that moves and shoots
- projectile - image of fired projectile
- sound - sound to play when firing projectile
- screen - screen to draw ships and projectiles on
- delta_time - delta time variable for smoother animation
- height - height of screen

4.5.1.2 boss_shoot_and_move

A method that updates boss movement and shooting. Takes the following arguments:

- time - time of method call
- projectile1 - image for projectile 1
- projectile2 - image for projectile 2
- sound - sound to play when firing projectile
- delta_time - delta time variable for smoother animation
- screen - screen to draw boss and projectiles on
- height - height of screen

4.5.1.3 enemy_destroyed

A method that removes destroyed enemy ships. Takes the following arguments:

- enemy_group - group of hit enemy ships
- audio_play - boolean value for playing audio or not
- sound_play - audio file to be played
- enemy_type - type of destroyed enemy. Used for scoring points. Use "advanced" to score 50 points or "regular" to score 10.
- explosion_anim - explosion animation to play
- explosion_anim_big – explosion animation for advanced enemy
- explosion_sound_big - audio file for big explosion
- explosion_sound_small - audio file for small explosion
- Explosion_class - class for explosion animation(e.g. Explosion())

4.5.1.4 enemy_advanced_damaged

A method that tracks damage to advanced enemy ships and destroys them if their health drops to 0. Takes the following arguments:

- enemy_hit_group – group of collided objects
- enemy_kill_group – group of killed ships
- enemy_group – group of enemy ships

4.5.1.5 boss_hit

A method that resolves laser bolts/torpedos that hit the boss. Takes the following arguments:

- hit_group - group of boss(e.g. self.enemy_boss_group)
- sound_play - boolean if sound is muted or not
- music_play - boolean if music is muted or not
- dmg - how much damage the boss takes from this hit
- explosion_sound - audio file to play if boss is destroyed
- music_track - music file to play is boss is destroyed
- explosion_class - class for explosion animation(e.g. Explosion())

4.5.1.6 torpedo_hit

A method that resolves torpedo hits. Takes the following arguments:

- music_track - music file to play is boss is destroyed
- explosion_anim - explosion animation to play
- explosion_anim_big – explosion animation for advanced enemy
- explosion_sound_big - audio file for destroyed advanced enemies and boss
- explosion_sound_small - audio file for destroyed basic enemies
- Explosion_class - class for explosion animation(e.g. Explosion())

4.5.1.7 menu_animations

Method that draws and updates idle animations in the menu. Takes the following arguments:

- screen - screen to draw on
- img - background to draw on screen

4.5.1.8 init_menu

A method that sets idle background animation for the different menu screens. Takes the following arguments:

- speed_front - speed value for the front image of the background animation
- speed_back - speed value for the back image of the background animation

4.5.1.9 new_game

The main game loop. This method takes no arguments and is divided into the following parts:

- Draw and render the background
- Draw player projectiles
- Draw and animate the player ship
- Draw and animate enemy ships, projectiles and explosions
- Check if an enemy is hit by laser
- Check if an enemy is hit by torpedo
- Check if an enemy collided with player ship or if an enemy killed player ship
- Check time before starting to spawn enemies. Check if enemies are less than max number of enemies and add more - adjust game difficulty in this section by changing the formula for max spawned enemies on screen
- Check kill count and spawn an advanced enemy for every 10 regular enemies killed - adjust advanced enemy spawn criteria here by changing the required number of basic enemies destroyed to spawn an advanced enemy
- Check for events
- Keypress events - change controls for player movement, fire, menu in this section
- Pause / Unpause - change pause menu button in this section
- Award lives and torpedoes on milestones - adjust player rewards(bonus torpedoes and lives) in this section
- Check boss attributes and update - most of the boss related attributes are
- Check points and spawn a boss every 3000 points - change boss spawn requirement in this section
- Update background and groups
- Render the player attributes
- Draw the icons and text
- Blit boss health separately, to not be drawn under other images
- Scale, update the screen and get ticks
- Check if player group is empty and add draw new player ship sprite if player has lives left
- Pause menu - separate loop for pause menu - adjust pause menu behavior here

4.5.1.10 sub_menu

Method to display static menu. Used for Instructions and Credits screens of the Main Menu. Takes the following arguments:

- picture - image to display over the background

4.5.1.11 game_over

A method called when the player has no more lives. Here the player inputs their initials and their highscore is saved.

4.5.1.12 highscores

A method that displays the current seven highscores, sorted in descending pattern.

4.5.1.13 main_menu

A method for displaying the Main Menu of the game. All other menu methods are called within this method's loop.

4.5.2 File parameters

Parameters used in the game are defined here, divided into the following sections:

- Initializing - initialization of needed modules
- Game global parameters - set global game parameters here
- Mute/unmute variables - booleans for music/sound
- Animation sprites - define animation sprites here
- Game audio assets - define music and sounds here
- Game images - define used images in the game here