SE 3XA3: Software Requirements Specification
Space Shooter

Team #105, Space Shooter
Hongzhao Tan, tanh10
Nishanth Raveendran, raveendn
Dananjay Prabaharan, prabahad

April 7, 2020

Table 1: **Revision History**

Date	**Version**	**Notes**
March 13, 2020	1.0	Initial Draft
April 6, 2020	2.0	Rev1 Update - Added exceptions to internal functions, added Alien class, and updated the introduction to the document.

This Module Interface Specification (MIS) document contains modules, types and methods for implementing the state of a game of Space Shooter. The Space Shooter application is a recreation of the 1980's "Space Invader" arcade game. The goal within the game is to avoid the spaceship from getting hit by upcoming obstacles such as meteors/aliens.

Constants Module

Module

Constants

Uses

N/A

Syntax

Exported Constants

WIDTH = 480
HEIGHT = 600
FPS = 60
POWERUP_TIME = 5000
BAR_LENGTH = 100
BAR_HEIGHT = 10
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)
RED = (255, 0, 0)
GREEN = (0, 255, 0)
BLUE = (0, 0, 255)
YELLOW = (255, 255, 0)

Semantics

State Variables

None

In-Game Assets Module

Module

InGameAssets

Uses

Constants

Syntax

Exported Constants

img_dir = path for the operating system to access the folder 'assets'

img_dir = path for the operating system to access the folder 'sounds'

explosion_anim = Dictionary that stores the animations of explosion of an enemy when a bullet hits the enemy, explosion of the enemy when a the in-game space ship hits the enemy, and explosion of the in-game space ship when an enemy hits the in-game space ship when the value of the shield on space ship equal to 0

all_sprite = An object ~~(of pygame.sprite.Group type)~~ that collects all of the game objects that need to be displayed on the user interface during the "Playing Game State"

bullets = An object ~~(of pygame.sprite.Group type)~~ that collects the game objects(Bullet type) representing the bullets that are fired from the in-game space ship due to users' request and the value of power of the space ship and need to be displayed on the user interface during the "Playing Game State"

powerups = An object ~~(of pygame.sprite.Group type)~~ that collects the game objects representing the powerups (Pow type) that need to be displayed on the user interface during the "Playing Game State"

mobs = An object ~~(of pygame.sprite.Group type)~~ that collects the game objects (Mob type) that need to be displayed on the user interface during the "Playing Game State"
shooting_sound = An object ~~(of pygame.mixer.Sound type)~~ that records the data of the sound which need to be played based on the audio stored in pew.wav file when the in-game space ship shoots bullets

missile_sound = An object ~~(of pygame.mixer.Sound type)~~ that records the data of the sound which need to be played based on the audio stored in rocket.ogg file when the in-game space ship shoots missiles

expl_sounds = A list ~~of pygame.mixer.Sound type~~ that collects the sounds which need to be played based on the audio stored in file expl3.wav and expl6.wav when any of the Mob type game objects collide on Player type or Bullet type or Missile type of game objects.

player_die_sound = An object ~~(of pygame.mixer.Sound type)~~ that records the data of the sound which need to be played based on the audio stored in rumble1.ogg file when the Player type game object collide on Mob type game objects when its value of shield equal to 0.

player_img = An object ~~of pygame.image type~~ that records the data of the image stored in playerShip1_orange.png file which represents the appearance of the Player type game object on user interface

player_mini_img = An object of ~~pygame.transform type~~ that records the data of the image that has been transformed from player_img by scaling the player_img by a proper proportion that suits the dimensions of the user interface

bullet_img = An object ~~of pygame.image type~~ that records the data of the image stored in laserRed16.png file which represents the appearance of the Bullet type game objects on user interface

missile_img = An object ~~of pygame.image type~~ that records the data of the image stored in missile.png file which represents the appearance of the Missile type game objects on user interface

meteor_images = A list ~~of pygame.image type~~ that records the data of the images represents the appearance of the Mob type game objects on user interface.

powerup_images = A dictionary that records the data of the images represents the appearance of the Pow type game objects on user interface.

font_name = An object of ~~pygame.font type~~ that records the data of the font of the text which would be displayed on the user interface

background = An object of ~~pygame.image type~~ that records the data of the images represents the appearance of the background of the user interface

Exported Access Program

None

Semantics

Environment Variables

assets = folder that stores all of the image files that are necessary to the system
assets = folder that stores all of the audio files that are necessary to the system
M_explosion_anim_imgs = files that stores all of the image files that are necessary for animation of the explosion of the Mob type game objects
P_explosion_anim_imgs = files that stores all of the image files that are necessary for the animation of the explosion of the Player type game objects
shoot_sound_file = file that stores the audio for the sound of shooting bullets
missile_sound_file = file that stores the audio for the sound of shooting missiles
expl_sound1, expl_sound2 = files that store the audio for the sound of explosion of Mob type game objects
sound_of_death = file that stores the audio for the sound of explosion of the death of Player type game object
player_img_file = file that stores the image for the appearance of Player type game object on user interface
bullet_img_file = file that stores the image for the appearance of Bullet type game object on user interface
missile_img_file = file that stores the image for the appearance of Missile type game object on user interface
'meteorBrown_big1', 'meteorBrown_big2', 'meteorBrown_med1', 'meteorBrown_med3', 'meteorBrown_small1', 'meteorBrown_small2', 'meteorBrown_tiny1' = files that store the image for the appearance of Mob type game objects of different sizes on user interface

fasterShoot_img_file= file that stores the image for the appearance of Pow type game objects (for faster shooting powerups) on user interface
doubleShoot_img_file= file that stores the image for the appearance of Pow type game objects (for double shooting powerups) on user interface
background_img_file = file that stores the image for the appearance of background of user interface

Game Objects Module

Module

GameObjects

Uses

InGameAssets
Constants

Exported Types

Bullet: Object
Explosion: Object
Missle: Object
Mob: Object
Player: Object
Pow: Object

Bullet

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Bullet	integer, integer	Bullet	None
update			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

new Bullet(x, y):

- input: x - x-coordinate of bullet, y - y-coordinate of bullet
- transition: Initializes all attributes of the Bullet object to the default settings to start the game.
- output: Bullet - The newly initialized Bullet object
- exception: None

update():

- input:
- transition: Updates the location of the inputted bullet based on the current game state.
- output: None
- exception: None

Explosion

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Explosion	integer, integer	Explosion	None
update			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

new Explosion($center, size$):

- input: center - coordinate for explosion to occur, size - size of the explosion
- transition: Initializes all attributes of the Explosion object to the default settings to start the game.
- output: Explosion - The newly initialized Explosion object
- exception: None

update():

- input:
- transition: Updates the screen with the explosion animation given the location and size.
- output: None
- exception: None

Missile

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Missile	integer, integer	Missile	None
update			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

new Missile(x, y):

- input: x - x-coordinate of missile, y - y-coordinate of missile
- transition: Initializes all attributes of the Missile object to the default settings to start the game.
- output: Missile- The newly initialized Missle object
- exception: None

update():

- input:
- transition: Updates the location of the inputted missile based on the current game state.
- output: None
- exception: None

Mob

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Mob		Mob	None
rotate			None
update			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

new Mob():

- input: None
- transition: None
- output: Mob - The newly initialized Mob object
- exception: None

rotate():

- input: None
- transition: Rotates the mob object every 50 milliseconds.
- output: None

- exception: None

update():

- input: None
- transition: Updates the position attributes of the mob object, and re-positions the mob to a random start position if it goes out of bounds.
- output: None
- exception: None

Alien

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Alien		Alien	None
update			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

new Alien():

- input: None
- transition: None

- output: Alien - The newly initialized Alien object
- exception: None

update():

- input: None
- transition: Updates the position attributes of the alien object, and re-positions the alien to a random start position if it goes out of bounds.
- output: None
- exception: None

Player

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Player		Player	None
update			None
shoot			None
powerup	integer		IndexError
hide			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

Player():

- input: None
- transition: Initializes all attributes of the Player object to the default settings to start the game.
- output: Player - The newly initialized Player object
- exception: None

update():

- input: None
- transition: Updates the position and state of the player. The state includes if it has a power up, and if it is shooting.
- output: None
- exception: None

shoot():

- input: None
- transition: Determines the type of shot based on power up, and shoots a bullet object from the player spaceship.
- output: None
- exception: None

powerup(i):

- input: i - the index for the corresponding power-up
- transition: Updates the power up state of the player. Based on the value of i the power-up state will change: 1 = default, 2 = double bullets, 3 = missle, 4 = rapid fire.
- output: None
- exception: IndexError - if an invalid power up index is inputted

hide():

- input: None
- transition: Hides the player object.
- output: None
- exception: None

Pow

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
new Pow	integer	Pow	None
update			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

new Pow(*center*):

- input: center - coordinate for power-up to load on
- transition: Initializes all attributes of the Power-up object to the default settings to start the game.
- output: Pow- The newly initialized Pow object

- exception: None

update():

- input:
- transition: Updates the location of the power-up based on the current game state.
- output: None
- exception: None

Game Functions Module

Module

GameFunctions

Uses

InGameAssets
Constants
GameObjects

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
main_menu			None
pause_menu			None
instruction_page			None
draw_text	screen, string, integer, integer, integer		NullReferenceException
draw_shield_bar	screen, integer, integer, integer		NullReferenceException
draw_lives	screen, integer, integer, integer, .png		NullReferenceException
newmob			None
newalien			None

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

main_menu():

- input: None
- transition: Loads respective menu screen and music based on the keys pressed on keyboard.
- output: None
- exception: None

pause_menu():

- input: None
- transition: Loads respective menu screen and music based on the keys pressed on keyboard.
- output: None
- exception: None

instruction_page():

- input: None
- transition: Loads instruction page screen and music based on the keys pressed on keyboard.
- output: None
- exception: None

draw_text($surf, text, size, x, y$):

- input: surf - surface of screen to display text, text - text to display, size - font size, x - x-coordinate to display text, y - y-coordinate to display text
- transition: Selects a cross-platform text to display the text on the device the game is opened on.
- output: None
- exception: NullReferenceException - Passing in Null value for the input parameters

draw_shield_bar($surf, x, y, pct$):

- input: surf - surface of screen to display shield bar, x - x-coordinate to display text, y - y-coordinate to display text, pct - current health level of spaceship
- transition: Draws the shield bar based on the current health level of the spaceship.
- output: None
- exception: NullReferenceException - Passing in Null value for the input parameters

draw_lives($surf, x, y, lives, img$):

- input: surf - surface of screen to display lives, x - x-coordinate to display text, y - y-coordinate to display text, lives - number of lives remaining, img - image to display for each live
- transition: Draws the amount of lives left for the spaceship based on the current game situation.
- output: None
- exception: NullReferenceException - Passing in Null value for the input parameters

newmob():

- input: None
- transition: Constantly adds meteors during the game.
- output: None
- exception: None

newalien():

- input: None
- transition: Constantly adds aliens during the game.
- output: None
- exception: None

Game State Module

Module

GameState

Uses

GameObjects
GameFunctions

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
display_update_gamestate			RuntimeError

Semantics

State Variables

running: bool
menu_display: bool

Environment Variables

None

Assumptions & Design Decisions

- The Game State module invokes objects and functions from other modules and has no functions of its own besides "display_update_gamestate()". This design choice allows for information hiding.
- The module runs a game loop which contains all game functionality, and constantly updates checks user key inputs, updates the states of all game objects (including spaceship), and graphically updates the game.

Access Routine Semantics

display_upadate_gamestate():

- input: None
- transition: Intializes graphics and sounds, then runs the "game loop" which constantly updates based on the user input and status of the game. The loop manages the menu operations of starting the game, viewing instructions, pausing the game, and quiting. The loop manages game mechanics which updates graphics, check for spaceship colliding with other objects, and updating health, score, and power ups.
- output: None
- exception: RuntimeError - when there are errors caused during the duration of the playing game state