SE 3XA3: Module Internal Specification
Spaceshooter Remix

Team #4, IRS Development
Ibrahim Malik maliki2
Ryan Schnekenburger schneker
Saad Khan khans126

December 5, 2018

Contents

List of Tables

List of Figures

Table 1: Revision History

Date	**Version**	**Notes**
November 5, 2018	Rev 0	Authored by Ibrahim, Saad, Ryan
December 5, 2018	Rev1	Updated document and added the changes recommended by TA.

1 Module Hierarchy

This section provides an overview of the module design. Modules are summarized in a hierarchy decomposed by secrets in Table 2. The modules listed below, which are leaves in the hierarchy tree, are the modules that will actually, be implemented.

M1: Hardware-Hiding Module

M2: Behaviour-Hiding Module

M3: Software Decision Module

Level 1	**Level 2**
Hardware-Hiding Module	
Behaviour-Hiding Module	Player_Control module Player_Hide module Player_Move module Player_Shoot module Player_Ability module Overheat_Contol module spawns module Asteroids module constants module
Software Decision Module	Animations Module IRS_Space_Shooter Module

Table 2: Module Hierarchy

2 MIS of the Asteroids Module

2.1 Uses

N/A

2.2 Syntax

2.2.1 Exported Types

2.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	-	-
update	-	-	-
rotate	-	-	-

2.3 Semantics

2.3.1 State Variables

original : image
spin : int
rotspeed : int
speed_y : int
speed_x : int
rect : image
radius : int
rect.x : int
rect.y : int
rect.top : int
rect.left: int
rect.right : int

2.3.2 Environmental variables

Screen: Display device
Speakers: Device speakers

2.3.3 Assumptions

The assumption is that the game constructor is already created and the asteroids are called upon after the user initiates the game.

2.3.4 Access Routine Semantics

init():

- transition: This generates movements for all sprites and objects and defines the initial state of rotation, the speed of objects, spins and the area of movement within the screen.

- exception: None

update():

- transition: This will update call on the rotate function to generate a new rotated position for the original asteroid and it will create new objects to appear on the screen as certain asteroids reach the edge of the screen. As an asteroid leaves the screen area, a new one is created in its place.
- exception: None

rotate():

- transition: This function creates a new image of the object when called upon in the exact same location of the original image except it is slightly rotated by a certain value.
- exception: None

3 MIS of the Spawns Module

3.1 Uses

N/A

3.2 Syntax

3.2.1 Exported Types

3.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
spawn_bullet	-	-	-
update	-	-	-
spawn_missle	-	-	-
update	-	-	-
spawn_powerup	-	-	-
update	-	-	-

3.3 Semantics

3.3.1 State Variables

speed_y : int
speed_x : int
image : missile_image, powerup_image, bullet_image
radius : int
rect.x : int
rect.y : int
rect.top : int
rect.centerx : int
rect.centery : int
rect.left: int
rect.right : int

3.3.2 Environmental variables

Screen: Display device
Speakers: Device speakers

3.3.3 Assumptions

The assumption is that the game constructor is already created and the spawns of in-game objects are generated after the user initiates the game.

3.3.4 Access Routine Semantics

spawn_bullet:

- transition: This creates the sprite for the bullets in the place of the position of the player sprite. It also defines the speed with which the bullet is released and the image to use for this bullet.
- exception: None

update():

- transition: Once the bullet has crossed the height of the screen, it will be destroyed. It also specifies that the bullet will continue to move across the screen upwards until it has cleared the height of the screen.
- exception: None

spawn_missile:

- transition: This creates the sprite for the missile in the place of the position of the player sprite. It also defines the speed with which the missile is released and the image to use for this missile
- exception: None

update():

- transition: Once the missile has crossed the height of the screen, it will be destroyed. It also specifies that the missile will continue to move across the screen upwards until it has cleared the height of the screen.
- exception: None

spawn_powerup:

- transition: This creates a sprite for powerups. In this revision 0, there are 2 powerups. One is a shield and one is a level-up. A random choice chooses one of the two and generates it while assigning a speed for it to come down the screen with and it creates the object in the middle of the screen.
- exception: None

update():

- transition: This specifies that the powerup object will continue to move across the screen downwards.
- exception: None

4 MIS of the Animations Module

4.1 Uses

constants

4.2 Syntax

4.2.1 Exported Types

4.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
main_menu	-	-	-
game_over	int, int	-	-
put_health_bar	Screen, int, int, int	-	INVALID_ARGUMENT
put_overheat_bar	Screen, int, int, int	-	INVALID_ARGUMENT
put_text	Screen, String, int, int, int	-	INVALID_TYPE
put_lives	Screen, int, int, int, pre-loaded png file	-	INVALID_ARGUMENT

4.3 Semantics

4.3.1 State Variables

All variables come from the constants module

4.3.2 Environmental variables

Screen: Display device Speakers: Device Speakers

4.3.3 Assumptions

The assumption is that the game constructor is already created.
main_menu():

- transition: Load song "menu.ogg" and play it through speakers. Load image "main.png" scale it to fit the length and height of the window listed in constants.Add the image to the screen.
  put_text(Screen, "PRESS RETURN TO START", 30, WIDTH/2, HEIGHT/2)
  put_text(Screen, "PRESS Q TO QUIT", 30, WIDTH/2, HEIGHT/2+40)
  event.type = KEYDOWN $\longrightarrow$ KEYDOWN = [RETURN] $\hookrightarrow$ continue $\vee$
  KEYDOWN = [Q] $\hookrightarrow$ quit
  Load song "ready.ogg" and play it through speakers. Display a fully blue screen.
  put_text(screen, "GET READY", 40, WIDTH/2, HEIGHT/2)

- exception: None

game_over():

- transition: Load song "menu.ogg" and play it through speakers. Load image "black.png" scale it to fit the length and height of the window listed in constants. Add the image to the screen.
  put_text(Screen, "GAME OVER!", 60, WIDTH/2, HEIGHT/4)
  put_text(Screen, "PRESS R To Return to Menu", 60, WIDTH/2, HEIGHT/4+60)
  if newhighscore then put_text(con.screen, "New High Score: " + str(highscore), 60, con.WIDTH/2, con.HEIGHT/4 + 120)
  event.type = KEYDOWN $\longrightarrow$ KEYDOWN = r $\hookrightarrow$ break $\vee$
  KEYDOWN = [Q] $\hookrightarrow$ quit

- exception: None

put_health_bar(surf, x, y, pct):

- transition: draw on the surf a rectangle x across the surf and y up the surf of length HEALTH_HEIGHT and width HEALTH_LENGTH with a white outline of thickness = 2. Fill the rectangle with a green box of height HEALTH_HEIGHT and length pct/100 • HEALTH_LENGTH.

- exception: $0 < \text{pct} \wedge \text{pct} > 100 \implies$ INVALID_ARGUMENT

put_overheat_bar(surf, x, y, pct):

- transition: draw on the surf a rectangle x across the surf and y up the screen of length OVERHEAT_HEIGHT and width OVERHEAT_LENGTH with a white outline of thickness = 2. Fill the rectangle with a red box of height OVERHEAT_HEIGHT and length pct/100 • OVERHEAT_LENGTH.

- exception: $0 < \text{pct} \wedge \text{pct} > 100 \implies$ INVALID_ARGUMENT

put_text(surf, text, size, x, y):

- transition: Display white text on the surf of characters in text, size of size, and a starting position of x across the surf and y up the surf.

- exception: T : text $\neq$ String $\implies$ INVALID_TYPE

put_lives(surf, x, y, lives, img):

- transition: Display the image lives times spaced 30 pixels apart where the first image is displayed x across the surf and y up the surf and all other images are displayed on the left.

- exception: $0 < \text{lives} \wedge \text{lives} > 3 \implies$ INVALID_ARGUMENT

Deleted this module

5 MIS of the Player Module

Split into more modules

5.1 Uses

constants, spawns

5.2 Syntax

5.2.1 Exported Types

5.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	set of sprites, set of sprites	Player	-
update	-	-	OUT_OF_BOUNDS_ERROR
shoot	-	-	-
ability	-	-	-
hide	-	-	-

5.3 Semantics

5.3.1 State Variables

image : image
radius : int
rect : Rect $\implies$ from pygame website
speedx : int
hot : boolean
health : int
overheat: int
shoot_delay : int
last_shot : real
lives : int
hidden : boolean
hide_timer : real
bullet : int
bullet_timer : real
mouse_down : boolean
all_sprites : set of sprites
bullets : set of sprites

5.3.2 Environmental variables

Screen: Display device
Speakers: Device Speakers

5.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

5.3.4 Access Routine Semantics

init(all_sprites, bullets):

- transition:
  image $\implies$ scaled player_img to 50 by 38
  remove black background from the image
  rect $\implies$ the Rect object of image
  set the midpoint of the rectangle's x coordinate to half of the width of the screen
  set the bottom of the rectangle to be the height of the screen - 10
  speedx $\implies$ 0
  hot $\implies$ False
  health $\implies$ 100
  overheat $\implies$ 0
  shoot_delay $\implies$ 250
  last_shot $\implies$ current time in milliseconds
  lives $\implies$ 3
  hidden $\implies$ False
  hide_timer $\implies$ current time in milliseconds
  bullet $\implies$ 1
  bullet_timer $\implies$ current time in milliseconds
  mouse_down $\implies$ False
  all_sprites $\implies$ all_sprites
  bullets $\implies$ bullets

- exception: None

shoot():

- transition:
  if bullet $\geq 2 \wedge$ the current time $-$ bullet_timer $>$ bulletUP_TIME $\longrightarrow$ bullet is decremented $\wedge$ bullet_time $\implies$ current time.

  if hidden = True $\wedge$ the current time $-$ hide_timer $> 1000 \longrightarrow$ hidden $\implies$ False

$\wedge$ set the midpoint of rect to half the width of the screen $\wedge$ set the bottom of the rectangle to the height of the screen - 30

speedx $\implies$ 0

LEFT ARROW is pressed speedx $\implies$ -5
RIGHT ARROW is pressed speedx $\implies$ 5

hot $\longrightarrow$ overheat is decremented, overheat = 0 $\longrightarrow$ hot = False
$\vee$
LEFT CLICK is pressed $\longrightarrow$ mouse_down = False $\wedge$ overheat $< 90 \wedge$ hidden = False $\longrightarrow$ shoot(), mouse_down $\implies$ True, overheat incremented by 20, overheat $> 90 \longrightarrow$ hot $\implies$ True
$\vee$
LEFT CLICK is not pressed $\longrightarrow$ mouse_down $\implies$ False, overheat $\neq 0 \longrightarrow$ overheat is decremented, overheat = 0 $\longrightarrow$ hot $\implies$ False
$\vee$
pass

rect moves to the right by speedx

- exception: The right-most coordinate of rect $>$ width $\vee$ The left-most coordinate of rect $< 0 \implies$ OUT_OF_BOUNDS_ERROR

update():

- transition:
  if bullet $\geq 2 \wedge$ the current time $-$ bullet_timer $>$ ABILITYUP_TIME $\longrightarrow$ bullet is decremented $\wedge$ bullet_time $\implies$ current time.

  if hidden = True $\wedge$ the current time $-$ hide_timer $> 1000 \longrightarrow$ hidden $\implies$ False $\wedge$ set the midpoint of rect to half the width of the screen $\wedge$ set the bottom of the rectangle to the height of the screen - 30

  speedx $\implies$ 0

  LEFT ARROW is pressed speedx $\implies$ -5
  RIGHT ARROW is pressed speedx $\implies$ 5

  hot $\longrightarrow$ overheat is decremented, overheat = 0 $\longrightarrow$ hot = False
  $\vee$
  LEFT CLICK is pressed $\longrightarrow$ mouse_down = False $\wedge$ overheat $< 90 \wedge$ hidden = False $\longrightarrow$ shoot(), mouse_down $\implies$ True, overheat incremented by 20, overheat $> 90 \longrightarrow$ hot $\implies$ True

$\vee$
LEFT CLICK is not pressed $\longrightarrow$ mouse_down $\implies$ False, overheat $\neq 0 \longrightarrow$ overheat is decremented, overheat $= 0 \longrightarrow$ hot $\implies$ False
$\vee$
pass

rect moves to the right by speedx

- exception: The right-most coordinate of rect $>$ width $\vee$ The left-most coordinate of rect $< 0 \implies$ OUT_OF_BOUNDS_ERROR

shoot():

- transition:
current time - last_shot $>$ shoot_delay $\longrightarrow$
bullet $= 1 \longrightarrow$
spawn a bullet a the top and the center of rect and add them to all_sprites and bullets
$\vee$
bullet $= 2 \longrightarrow$
spawn 2 bullets one to the left of the center of rect and another to the right of the center of rect and add them to all_sprites and bullets
$\vee$
bullet $= 3 \implies$
spawn 2 bullets one to the left of the center of rect and another to the right of the center of rect. Spawn a missile on the center of rect and the top of rect. Add all three to bullets and all_sprites.

- exception: None

ability():

- transition:
increment bullet and update bullet_timer to the current time

- exception: None

hide():

- transition:
hidden $\implies$ True, update hide_timer to the current time, update the center of rect to WIDTH / 2 in the x and HEIGHT + 200 in the y

- exception: None

Additions Start here

6 MIS of the Player_Control Module

6.1 Uses

Player_Shoot, Player_Hide, Player_Ability, Player_Move, Overheat_Control, constants, spawns,

6.2 Syntax

6.2.1 Exported Types

T : player

6.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	array of sprites, array of sprites	Player	OUTOFBOUNDS
update	-	-	-
ability	-	-	-
update_Health	int	-	-
reset_Health	-	-	-
update_Lives	int	-	-
get_hide	-	boolean	-
get_Overheat	-	int	-
get_Lives	-	int	-

6.2.3 State Variables

image : image
rect : Rect
health : int
lives : int
sprites : array of sprites
bullets : array of sprites
pos : move hide : hidden ovrht : Overheat shot : shoot power : ability

6.2.4 Environmental variables

Screen: Display device
Speakers: Device Speakers

6.2.5 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

6.2.6 Access Routine Semantics

init(sprites, bullets):

- transition:
  image $\implies$ player sprite
  rect $\implies$ rectangular outline of image
  center of rect $\implies$ WIDTH/2
  bottom of rect $\implies$ HEIGHT - 10
  health $\implies$ 100
  sprites $\implies$ sprites
  bullets $\implies$ bullets

- exception: None

noindent update():

- transition:
  power.get_Power ≥ 2 $\implies$ power.powerdown
  hide.get_hide $\implies$ hide.unhide(rect) pos.reset_Speed
  if left click $\implies$ shot.det_shoot_T(sprites, bullets, rect, hide, ovrht, power.get_Power())
  if not left click $\implies$ shot.det_shoot_F(ovrht)
  if a $\implies$ pos.move_Left
  if d $\implies$ pos.move_Right

- exception: if player is at position ¿ WIDTH or player is at position ¡ width *implies* OUTOFBOUNDS

ability():

- transition:
  power.powerup()

- exception: None

update_Health(aid):

- transition:
  health $\implies$ health + aid

- exception: None

reset_Health(aid):

- transition:
  health $\Longrightarrow$ 100

- exception: None

update_Lives(life):

- transition:
  lives $\Longrightarrow$ lives + life

- exception: None

get_Overheat():

- transition:
  return: ovrht.get_Overheat()

- exception: None

get_Health():

- transition:
  return: health

- exception: None

get_hide():

- transition:
  return: hide.get_hide()

- exception: None

get_Lives():

- transition:
  return: lives

- exception: None

die():

- transition:
  hide.hide(rect)
  lives $\implies$ lives - 1
  health $\implies$ 100
  ovrht.reset_Overheat()
  ovrht.cool()

- exception: None

7 MIS of the Ability Module

7.1 Uses

constants

7.2 Syntax

7.2.1 Exported Types

T : ability

7.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	ability	-
powerdown	-	-	-
powerup	-	-	-
get_Power	-	int	-

7.3 Semantics

7.3.1 State Variables

power : int power_timer : int

7.3.2 Environmental variables

7.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

7.3.4 Access Routine Semantics

init():

- transition:
  power $\Longrightarrow$ 1
  power_timer $\Longrightarrow$ current time

- exception: None

powerdown():

- transition:
  if current time - power_timer > con.poweruptime $\Longrightarrow$ power becomes power - 1
  power_timer $\Longrightarrow$ current time

- exception: None

powerup():

- transition:
  power $\Longrightarrow$ power + 1
  power_timer $\Longrightarrow$ current time

- exception: None

get_Power():

- transition:
  return: power

- exception: None

8 MIS of the Move Module

8.1 Uses

8.2 Syntax

8.2.1 Exported Types

T : move

8.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	-	-
move_Left	-	-	-
move_Right	-	-	-
move_Player	rect	-	-
reset_Speed	-	-	-
get_Speed	-	speed_x	-

8.3 Semantics

8.3.1 State Variables

speed_x : int

8.3.2 Environmental variables

8.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

8.3.4 Access Routine Semantics

init():

- transition:
  speed_x $\implies$ 0

- exception: None

move_Left():

- transition:
  speed_x $\implies$ speed_x - 5

- exception: None

move_Right():

- transition:
  speed_x $\implies$ speed_x + 5

- exception: None

move_Player(rect):

- transition:
  rect.x $\implies$ rect.x + speed_x

- exception: None

reset_Speed():

- transition:
  speed_x $\implies$ 0

- exception: None

get_Speed():

- transition:
  return: speed_x

- exception: None

9 MIS of the Player_Hide Module

9.1 Uses

constants

9.2 Syntax

9.2.1 Exported Types

T : hidden

9.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	hidden	-
init	-	-	-
unhide	rect	-	-
hide	rect	-	-
get_hide	-	int	-

9.3 Semantics

9.3.1 State Variables

off : boolean
hidden_timer : int

9.3.2 Environmental variables

9.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

9.3.4 Access Routine Semantics

init():

- transition:
  off = False
  hidden_timer = current time

- exception: None

unhide(rect):

- transition:
  if current time - hide_timer > hidetime $\implies$ off becomes False and rect.center = WIDTH/2 and rect.bottom = HEIGHT - 30

- exception: None

hide(rect):

- transition:
  off *implies* True
  hide_timer $\implies$ current time
  rect.center = (WIDTH/2, HEIGHT + 200)

- exception: None

get_hide():

- transition:
  return: off

- exception: None

10 MIS of the Overheat_Control Module

10.1 Uses

constants, spawns

10.2 Syntax

10.2.1 Exported Types

T : Overheat

10.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	Overheat	-
get_Overheat	-	int	-
reset_Overheat	-	-	-
cool	-	-	-
update_Overheat	int	-	OUTOFBOUNDS
update_Hot	boolean	-	-
get_Hot	-	boolean	-

10.3 Semantics

10.3.1 State Variables

overheat : int
hot : boolean

10.3.2 Environmental variables

10.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

10.3.4 Access Routine Semantics

init():

- transition:
  overheat $\Longrightarrow$ 0
  hot $\Longrightarrow$ False

- exception: None

get_Overheat():

- transition:
  return: overheat

- exception: None

reset_Overheat():

- transition:
  overheat $\Longrightarrow$ 0

- exception: None

cool():

- transition:
  hot $\Longrightarrow$ False

- exception: None

update_Overheat(d0):

- transition:
  overheat $\Longrightarrow$ overheat + d0

- exception: if overheat < 0 then overheat becomes 0 or if overheat is $>$ than 100 overheat $= 100$

update_Hot(update):

- transition:
  hot $\Longrightarrow$ update

- exception: None

get_Hot():

- transition:
  return: Hot

- exception: None

11 MIS of the Player_Shoot Module

11.1 Uses

spawns, constants, Overheat_Control, Player_Hide

11.2 Syntax

11.2.1 Exported Types

T : shoot

11.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	shoot	-
det_shoot_T	array of sprites, array of sprites, rect, Hidden, Overheat, int	-	-
det_shoot_F	Overheat	-	-
fire	rect, array of sprites, array of bullets, power	-	-

11.3 Semantics

11.3.1 State Variables

mouse_down : boolean

11.3.2 Environmental variables

Screen: Display device
Speakers: Device Speakers

11.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

11.3.4 Access Routine Semantics

init():

- transition:
  last_shot $\implies$ current time
  mouse_down $\implies$ False

- exception: None

det_shoot_T(sprites, bullets, rect, hide, ovrht, power):

- transition:
  if ovrht.get_Hot() then ovrht.update_Overheat(-1) and if ovrht.get_Overheat() = 0 then ovrht.update_Hot(False)
  or if mouse_down = False and ovrht.get_Overheat $<$ 90 and hide.get_hide() = False then mouse_down = True and ovrht.update_Overheat(20) and fire(rect, sprites, bullets, power) and
  if ovrht.get_Overheat() $>$ 90 then ovrht.update_Hot(True)
  or if hide.get_hide() then fire(rect, sprites, bullets, power) or ovrht.update_Overheat(-1)

  Word explanation if the overheat bar is hot then decrement it by one and check to see if it is now zero if it is set the overheat bar to cool and exit. If the mouse has not been pressed right before the shot and overheat is less than 90 and the player is not hidden the mouse_down is set to true, the overheat goes up by 20 and fire if the overheat bar is greater than 90 it is hot. If the player is hidden fire without overheat penalty and exit. If nothing above is true decrement the overheat bar by 1.

- exception: None

det_shoot_F(ovrht):

- transition:
  mouse_down $\implies$ False if ovrht.get_Overheat() $\neq$ 0 then ovrht.update_Overheat(-1) or ovrht.update_Hot(False)

- exception: None

fire(rect, sprites, bullets, power):

- transition:
  if power = 1 then spawn 1 bullet at the top center of the ship
  if power = 2 then spawn 2 bullets one at the left of the ship another at the right of the ship
  if power = 3 then spawn 2 bullets and one missile. The missile at the top center of the ship and the bullets in the same place as power = 2.
  if power $\geq$ 4 then spawn 3 missiles each in the same spawn points as in power = 3
- exception: None

Additions Stop here

12 MIS of the Destroy Module

12.1 Uses

constants

12.2 Syntax

12.2.1 Exported Types

T : Destroy

12.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	(int, int), int	Destroy	-
update	-	-	-

12.3 Semantics

12.3.1 State Variables

image : image
rect : Rect
cur_frame : int
frames : int
updated : real
size : int

12.3.2 Environmental variables

Screen: Display device
Speakers: Device Speakers

12.3.3 Assumptions

The assumption is that the game constructor is already created and the player is playing the game.

12.3.4 Access Routine Semantics

init(center, size):

- transition:
  size $\implies$ size
  image $\implies$ explosion.animation[size][0] rect $\implies$ rectangular borders of image center of rect becomes the inputted center coordinate
  frames $\implies$ 75
  updated $\implies$ current time
- exception: None

update():

- transition:
  current time - updated > frames $\longrightarrow$
  updated = current time, cur_frame is incremented by 1, current frame = length of the explosion animation is of the given size $\longrightarrow$ remove the sprite $\vee$
  image $\implies$ the explosion animation of the size at the current frame,
  center $\implies$ the center point of the rect,
  rect $\implies$ the rectangular borders of the new image,
  the center of the rect would then become center
- exception: None

13 MIS of the Constants Module

13.1 Uses

None

13.2 Syntax

13.2.1 Exported Types

13.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	-	-

13.3 Semantics

13.3.1 State Variables

WHITE : (int, int, int)
BLACK : (int, int, int)
RED : (int, int, int)
GREEN : (int, int, int)
BLUE : (int, int, int)
YELLOW : (int,int,int)
WIDTH : int
HEIGHT : int
screen : display
HEALTH_LENGTH : int
HEALTH_HEIGHT : int
OVERHEAT_LENGTH : int
OVERHEAT_HEIGHT : int
FPS : int
ABILITYUP_TIME : int
BAR_LENGTH : int
BAR_HEIGHT : int
images : path to file
sounds : path to file
font_name : Font
player_img : image
player_mini_img : image
explosion_animation : hash of array of int

13.3.2 Environmental variables

Screen: Display device
Speakers: Device Speakers

13.3.3 Access Routine Semantics

init(center, size):

- transition:
  WHITE : (255, 255, 255)
  BLACK : (0, 0, 0) RED : (255, 0, 0)
  GREEN : (0, 255, 0)
  BLUE : (0, 0, 255)
  YELLOW : (255, 255, 0)
  WIDTH : 480
  HEIGHT : 600
  screen : display of width WIDTH and HEIGHT height resizeable
  HEALTH_LENGTH : 100
  HEALTH_HEIGHT : 10
  OVERHEAT_LENGTH : 100
  OVERHEAT_HEIGHT : 10
  FPS : 60
  ABILITYUP_TIME : 5000
  BAR_LENGTH : 100
  BAR_HEIGHT : 10
  images : path to objects file
  sounds : path to sounds file
  font_name : Arial
  player_img : loaded image of 'playership1_orange.png'
  player_mini_img : scaled image of player_img by 25 by 19
  remove black background from player_mini_image
  explosion_animation : insert 9 explosions scaled by 75 by 75 to the first location, 9 explosions scaled by 32 by 32 in the second location, and 9 explosions to the third location. The indexing of the file location depend on the number in the file name.

- exception: None

14 MIS of the IRS Space Shooter Module

14.1 Uses

pygame, random, Animations, spawns, Asteroids, Player, Destroy

14.2 Syntax

14.2.1 Exported Types

14.2.2 Exported Access Programs

Routine name	In	Out	Exceptions
init	-	-	-

14.3 Semantics

14.3.1 State Variables

images : str
sounds : str
WHITE : (int, int, int)
BLACK : (int, int, int)
RED : (int, int, int)
GREEN : (int, int, int)
BLUE : (int, int, int)
YELLOW : (int,int,int)
WIDTH : int
HEIGHT : int
clock : int
background : img
player_img : img
player_mini_img : img
bullet_img : img
missle_img : img
meteor_images : img
explosion_animation : img
powerup_images : img
shooting_sound : sound
missle_sound : sound
expl_sound : sound
all_sprites : sprite
player : player
mobs : sprite

14.3.2 Environmental variables

Screen: Display device
Speakers: Device Speakers

14.3.3 Access Routine Semantics

- transition:
  images : path to objects file
  sounds : path to sounds file
  WHITE : (255, 255, 255)
  BLACK : (0, 0, 0) RED : (255, 0, 0)
  GREEN : (0, 255, 0)

BLUE : (0, 0, 255)
YELLOW : (255, 255, 0)
WIDTH : 480
HEIGHT : 600
clock : game clock
background : background image
player_img : player sprite
player_mini_img : scaled image of player
sprite to 25x19
bullet_img : bullet sprite
missle_img : missle sprite
meteor_images : array of meteor sprites
explosion_animation : array of explosion sprites
powerup_images : powerup sprites
shooting_sound :shooting sound bytes
missle_sound : missle sound bytes
expl_sound : explosion sound bytes
all_sprites : array of all sprites
player :instance of the player
mobs : array of asteroids

- exception: None