

SE 3XA3: Module Interface Specification T-Rex Acceleration

Team 15, Dev^{enthusiasts}

Zihao Du (duz12)

Andrew Balmakund (balmakua)

Namit Chopra (choprn9)

March 18, 2021

Table 1: **Revision History**

Date	Version	Notes
03/18/2021	1.0	Initial Draft

Model/Character Module

Module

Character

Uses

Pygame // Uses class Sprite
Time // calculates the duration of powerups

Syntax

Exported Constants

N/A

Exported Types

N/A

Exported Access Programs

Routine name	In	Out	Exceptions
character	pygame.display, String	Character	IllegalArgumentException, FileNotFoundError
set_img	pygame.image	—	IllegalArgumentException
duck	pygame.image, pygame.image	—	FileNotFoundError
stand	pygame.image, pygame.image	—	—
jump	—	—	—
invincible	pygame.image	—	FileNotFoundError
double_jumping	—	—	—
slo_mo	—	—	—
update	—	—	—

Semantics

State Variables

game_window: pygame.display	//	Represents the screen
img: pygame.image	//	Surface object with the image of the character
is_ducking: \mathbb{B}	//	Is the character ducking
is_jumping: \mathbb{B}	//	Is the character jumping
is_invincible: \mathbb{B}	//	Is the character invincible
is_slo_mo: \mathbb{B}	//	Is the character slowing obstacles
is_double_jumping: \mathbb{B}	//	Can the character double jump
movement: (\mathbb{R}, \mathbb{R})	//	horizontal and vertical speed of the character

Environment Variables

None

State Invariant

$is_ducking \wedge is_jumping = False$
 $is_invincible \wedge is_slo_mo \wedge is_double_jumping = False$

Assumptions

Constructor character is called first before other methods.

Access Routine Semantics

character(screen, img_path):

- transition: game_window, img, movement := screen, pygame.image.load(img_path), [0,0]
All Boolean state variables are initialized to False
- output: $out := self$
- exception:

$exc := (img \equiv \neg FileExists) \Rightarrow FileNotFoundError$

$exe := (game_window \equiv NULL) \Rightarrow IllegalArgumentError$

set_img(new_img):

- transition: $img := new_img$

- exception:

$exe := (new_img \equiv NULL) \Rightarrow IllegalArgumentError$

duck(ducking_img, inv_ducking_img):

- transition: $\neg is_jumping \Rightarrow (is_ducking := False;$
 $is_invincible \Rightarrow img := inv_ducking_img \wedge \neg is_invincible \Rightarrow img := ducking_img)$

- exception:

$exc := (img \equiv \neg FileExists) \Rightarrow FileNotFoundExceptionError$

stand(inv_char, char_img):

- transition: $\neg is_jumping \Rightarrow (is_ducking := False;$
 $is_invincible \Rightarrow img := inv_char \wedge \neg is_invincible \Rightarrow img := char_img)$

jump():

- transition: $\neg (is_ducking \wedge is_jumping) \Rightarrow (is_jumping, movement[1] := True, INIT_SPEED)$

invincible(inv_char):

- transition: $is_invincible, img, is_double_jumping, is_slo_mo := True, inv_char, False,$
 $False$

- exception:

$exc := (img \equiv \neg FileExists) \Rightarrow FileNotFoundExceptionError$

double_jump():

- transition: $is_double_jumping, is_slo_mo, is_invincible := True, False, False$

slo_mo():

- transition: $is_slo_mo, is_invincible, is_double_jumping := True, False, False$

update():

- transition: Change $is_double_jumping$ or $is_invincible$ or is_slo_mo to $False$ and the corresponding image back if the power up has already lasted for $DURATION_TIME$. If $is_jumping$ is $True$, decrease $movement[1]$ by $GRAVITY$, and check if the jumping ends(if the character back to ground)

Local Constants

INIT_SPEED = -20

GRAVITY = 1

DURATION_TIME = 5

Model/Powerups

Uses

Pygame // Uses class Sprite

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
powerups	$\mathbb{N}, \mathbb{N}, \mathbb{R}$, string	powerups	—
get_width	—	\mathbb{N}	—
set_width	\mathbb{N}	—	IllegalError
get_height	—	\mathbb{N}	—
set_height	\mathbb{N}	—	IllegalError
get_speed	—	\mathbb{R}	—
set_speed	\mathbb{N}	—	—
get_img	—	string	—
set_img	string	—	FileNotFoundError
get_name	—	String	—

Semantics

State Variables

name: String // Represents the type of a power up
width: \mathbb{N} // Represents the width of a power up
height: \mathbb{N} // Represents the height of a power up
speed: \mathbb{R} // Represents the speed of a power up
img: pygame.image // Surface object with a specified image drawn onto it

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

powerups(width, height, speed, imgPath):

- transition:

width, height, speed, img := width, height, speed, pygame.image.load(imgPath)
name = random element in POWERUP_LIST

- output: $out := self$

get_width():

- output:

out := width

set_width(new_width):

- transition:

width := new_width

- exception:

exc := (new_width < 0) \Rightarrow *IllegalArgumentException*

get_height():

- output:

out := height

set_height(new_height):

- transition:

height := new_height

- exception:

exc := (new_height < 0) \Rightarrow *IllegalArgumentException*

get_speed():

- output:

```

        out := speed
set_speed(new_speed):
    • transition:
        speed := new_speed
get_img():
    • output:
        out := img
set_img(img_path):
    • transition:
        img := pygame.image.load(img_path)
    • exception:
        exc := (imgPath  $\equiv \neg FileExists$ )  $\Rightarrow FileNotFoundError$ 
get_name():
    • output:
        out := name

```

Local Constants

```
POWERUP_LIST = ["Invincibility", "Double Jump", "Score Boost", "Slo_mo"]
```


Model/Obstacle

Uses

Vector2D

Pygame // Class for using pygame library (fixed definition)

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
obstacle	string, \mathbb{N} , \mathbb{N} , \mathbb{R} , string	Obstacle	—
get_width	—	\mathbb{N}	—
set_width	\mathbb{N}	—	IllegalError
get_height	—	\mathbb{N}	—
set_height	\mathbb{N}	—	IllegalError
get_speed	—	\mathbb{R}	—
set_speed	\mathbb{N}	—	—
get_img	—	string	—
set_img	string	—	FileNotFoundError
get_rect	—	(\mathbb{R}, \mathbb{R})	—
set_rect	(\mathbb{R}, \mathbb{R})	—	—

Semantics

State Variables

name: String // Represents the name of an obstacle
width: \mathbb{N} // Represents the width of an obstacle
height: \mathbb{N} // Represents the height of an obstacle
speed: \mathbb{R} // Represents the speed of an obstacle
img: pygame.image // Surface object with a specified image drawn onto it
rect: (\mathbb{R}, \mathbb{R}) // X and Y coordinates of the obstacle

State Invariant

None

Assumptions & Design Decisions

None

Access Routine Semantics

obstacle(name, width, height, speed, imgPath):

- transition:

name, width, height, speed, img := name, width, height, speed, pygame.image.load(imgPath)

rect := img.get_rect()

- output: *out* := *self*

get_width():

- output:

out := width

set_width(new_width):

- transition:

width := new_width

- exception:

exc := (new_width < 0) \Rightarrow *IllegalArgumentException*

get_height():

- output:

out := height

set_height(new_height):

- transition:

height := new_height

- exception:

exc := (new_height < 0) \Rightarrow *IllegalArgumentException*

get_speed():

- output:

```

    out := speed
set_speed(new_speed):
    • transition:
        speed := new_speed
get_img():
    • output:
        out := img
set_img(img_path):
    • transition:
        img := pygame.image.load(img_path)
    • exception:
        exc := (imgPath  $\equiv \neg FileExists$ )  $\Rightarrow FileNotFoundError$ 
get_rect():
    • output:
        out := rect
set_rect(x, y):
    • transition:
        rect.x := x
        rect.y = y

```

Model/DetectCollision

Uses

Character // Class for Characters
Powerups // Class for power ups
Obstacle // Class for Obstacles
Pygame

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
detect_collision	pygame.sprite, seq of pygame.sprite	\mathbb{B}	—
find_collision	pygame.sprite, seq of pygame.sprite	pygame.sprite	—

Semantics

State Variables

None

State Invariant

None

Assumptions & Design Decisions

Character and Obstacle have been defined, only comparing one obstacle with one character at a time.

Access Routine Semantics

detect_collision(Character, Elements):

- output: $\text{out} := \forall \text{element} \in \text{Elements} | (\text{Character.X} < \text{element.X} + \text{element.width}) \wedge (\text{Character.X} + \text{Character.width} > \text{element.X}) \wedge (\text{Character.Y} < \text{element.Y} + \text{element.height}) \wedge (\text{Character.Y} + \text{Character.height} > \text{element.Y})$

find_collision(Character, Elements):

- output: $\text{out} :=$ The element in the sequence that collides with Character

Model/UpdateEnvironment

Uses

Pygame

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
update_floor	pygame.image, \mathbb{Z} ,pygame.display	\mathbb{Z}	—
update_bg_colour	\mathbb{Z} , \mathbb{Z} , seq of \mathbb{Z}	seq of \mathbb{Z}	—

Semantics

State Variables

None

State Invariant

None

Assumptions

The game window and images are defined before any routines are called.

Access Routine Semantics

update_floor(floor, floor_position, screen):

- output: $out := (\text{floor_position} \leq -500) \Rightarrow \text{floor_position} = 0 \vee \text{floor_position} = \text{floor_position} - \text{MOVEMENT_SPEED}$

update_bg_colour(current_score, previous_score, bg_rgb, screen):

- output: $out :=$ Every time the score is a multiple of CHANGE_BG, the value of either red (bg_rgb[0]), green (bg_rgb[1]), or blue (bg_rgb[2]) changes. The new value of one of the field is: $(0 \leq x \leq 2 \mid (bg_rgb[x] + 50) \% 255)$

Local Constants

MOVEMENT_SPEED: 10

CHANGE_BG = 50

Model/Score

Uses

Time // Module for time related functions

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
Score	—	Score	—
get_score	—	\mathbb{N}	—
update_score	—	—	—
get_high_score	—	\mathbb{N}	—

Semantics

State Variables

high_score: \mathbb{N}
current_score: \mathbb{N}
previous_score: \mathbb{N}
start_time: \mathbb{R}

State Invariant

None

Assumptions

None.

Access Routine Semantics

Score():

- output: $out := self$
- transition: high_score, current_score, start_time := 0, current time.

get_score():

- output: $out := current_score$

update_score():

- output: $out := current_score, previous_score$
- transition: $previous_score, current_score, high_score := current_score, ((current\ time - start_time) \cdot SCALE_FACTOR)$ rounded to the nearest natural number, $(current_score > high_score) \Rightarrow (high_score = current_score)$

get_score():

- output: $out := high_score$

Local Constants

SCALE_FACTOR: 5

Model/MainMenu

Uses

None

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
MainMenu	—	MainMenu	—
change_volume	\mathbb{N} , \mathbb{N}	—	—
get_volumes	—	\mathbb{N} , \mathbb{N}	—

Semantics

State Variables

background_music_volume: \mathbb{N}

sound_effects_volume: \mathbb{N}

State Invariant

None

Assumptions

None

Access Routine Semantics

MainMenu():

- output: $out := self$
- transition: background_music_volume, sound_effects_volume := MAX_VOLUME, MAX_VOLUME

change_volume(new_background_volume, new_sound_effects_volume):

- transition: background_music_volume, sound_effects_volume := new_background_volume, new_sound_effects_volume

get_volumes():

- output: *out* := background_music_volume, sound_effects_volume

Local Constants

MAX_VOLUME: 100

View/DisplayObstacle

Uses

Pygame // Uses PyGame library
Time // Time will be used to keep track of time
Obstacle //
CharacterType //
Random // Use to generate random integers in a specified range
LoadAssests //

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayObstacle	pygame.display	DisplayObstacle	IllegalArgumentException
get_obstacle_list	—	seq of ObstacleType	—
remove_obstacle	Obstacle	—	—
generate_obstacle	seq of Obstacle, \mathbb{R}	\mathbb{R}	—
draw_obstacle	$\mathbb{R}, \mathbb{R}, \text{Obstacle}$	—	—
update_obstacle_display	—	—	—

Semantics

State Variables

game_window: pygame.display
obstacle_list: seq of ObstacleType

State Invariant

None

Assumptions

game_window \neq null

Access Routine Semantics

DisplayObstacle(game_screen):

- output: *out* := *self*

- transition: `game_window, obstacle_list := game_screen, []`
- exception: *exc*: `(game_screen = NULL \Rightarrow IllegalArgumentException)`

`get_obstacle_list()`:

- output: *out* := `obstacle_list`

`remove_obstacle(obstacle)`:

- transition: `obstacle_list := obstacle_list - obstacle`.

`draw_obstacle(currentX, currentY, obstacle)`:

- transition: `game_window := Draw the given obstacle at a specific location (X, Y) coordinates using the obstacle image.`

`generate_obstacle(type_of_obstacles, prevObstacleSpawnTime)`:

- transition: Generate a random kind of obstacle after `APPROPRIATE_TIME` and add it to the `obstacle_list`.
- out: *out* := current time

`update_obstacle_display()`:

- transition: For each obstacle in `obstacle_list`, draw each obstacle at its new position by considering the speed of each object. Once each obstacle has been drawn off the screen (obstacle is outside the boundaries of the `game_window`, remove the obstacle from the list).

Local Constants

`APPROPRIATE_TIME` = random integer between 3 and 5 seconds

View/DisplayPowerups

Uses

Pygame // Uses PyGame library
Powerups
DetectCollision
Random // Use to generate random integers in a specified range
LoadAssests //

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayPowerups	pygame.display	DisplayPowerups	IllegalArgumentException
generate_powerup	—	—	—
draw_powerups	powerups	—	—
update_powerup	—	—	—

Semantics

State Variables

powerups_diplayed: seq of Powerups
game_screen: pygame.display

State Invariant

None

Assumptions

game_window \neq null

Access Routine Semantics

display_powerup(game_screen):

- output: $out := self$
- transition: $game_screen, powerups_diplayed := game_screen, []$

- exception: *exc*: (`game_screen = NULL` \Rightarrow `IllegalArgumentException`)

`generate_powerup()`:

- transition: `powerups_displayed` := Add a random kind of powerup in the `powerups_displayed` (a list of powerups displayed on the `game_window`).

`draw_powerups(powerup)`:

- transition: draw the power up on the screen

`update_powerup()`:

- transition: All elements in `powerups_displayed` are updated by position and drawn.

View/DisplayEnvironment

Uses

UpdateEnvironment
Pygame
LoadAssests

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayEnvironment	pygame.display	—	IllegalArgumentException
draw_score	Score,	—	—
display_instruction	String	—	—
draw_floor	pygame.image \mathbb{Z}	—	—
draw_background	seq of \mathbb{N}	—	—

Semantics

State Variables

None

State Invariant

game_screen: pygame.display

Assumptions

None

Access Routine Semantics

DisplayEnvironment(window):

- transition: game_screen := window

draw_score(score):

- transition: The score is drawn on the screen.

`draw_instruction(instructions):`

- transition: The instruction is shown on the screen for `SOME_TIME` and disappears after.

`draw_floor(floor, floor_position, game_screen):`

- transition : Draw the floor onto the screen

Local Constants

`SOME_TIME` = 5 seconds

View/DisplayWindow

Uses

Pygame

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayWindow	—	DisplayWindow	—
get_game_screen	—	pygame.display	—

Semantics

State Variables

game_screen: pygame.display

State Invariant

None

Assumptions

None

Access Routine Semantics

DisplayWindow():

- output: $out := self$
- transition: $game_screen :=$ a new pygame window with dimensions $WIDTH \times HEIGHT$ pixels

get_game_screen():

- output: $out := game_screen$

Local Constants

WIDTH = 800

HEIGHT = 600

View/DisplayCharacter

Uses

None

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayCharacter	pygame.display, Character	—	—
draw_character	—	—	—

Semantics

State Variables

game_screen: pygame.display
game_character: Character

State Invariant

None

Assumptions

None

Access Routine Semantics

DisplayCharacter(window, character):

- transition: game_screen, game_character := window, character

View/PlaySound

Uses

LoadAssets

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
PlaySound	—	PlaySound	—
play_bg_music	—	—	—
play_jump_sound	—	—	—
play_duck_sound	—	—	—
play_collision_sound	—	—	—
play_powerup_sound	—	—	—

Semantics

State Variables

background_music: pygame.mixer
jump_sound: pygame.mixer
duck_sound: pygame.mixer
collision_sound: pygame.mixer
powerup_pickup_sound: pygame.mixer

State Invariant

None

Assumptions

None

Access Routine Semantics

PlaySound():

- output: $out := self$

- transition: background_music, jump_sound, duck_sound, collision_sound, powerup_pickup_sound
:= the assets are loaded in from the LoadAssets module.

play_bg_music():

- output: *out* := None
- transition: plays the background music.
- exception: None

play_jump_sound():

- output: *out* := None
- transition: plays the jump sound effect.
- exception: None

play_duck_sound():

- output: *out* := None
- transition: plays the duck sound effect.
- exception: None

play_collision_sound():

- output: *out* := None
- transition: plays the collision sound effect.
- exception: None

play_powerup_sound():

- output: *out* := None
- transition: plays the powerup pickup sound effect.
- exception: None

View/DisplayMenu

Uses

None

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
DisplayMenu	pygame.display	—	—
display_main_menu	—	—	—
display_pause_menu	—	—	—
display_exit_menu	—	—	—
display_setting_menu	—	—	—

Semantics

State Variables

game_screen: pygame.display

State Invariant

None

Assumptions

None

Access Routine Semantics

DisplayMenu(window):

- transition: game_screen := window

display_main_menu():

- transition: game_screen := Main menu with ‘Play’, ‘Quit’ and ‘Setting’ buttons

display_pause_menu():

- transition: `game_screen :=` Pause menu with ‘Resume’ button to resume back to the current game or ‘Exit’ the game to the main menu

`display_exit_menu()`:

- transition: `game_screen :=` Exit menu with ‘Return’ to the main menu after game session has ended

`display_setting_menu()`:

- transition: `game_screen :=` Setting menu that can change volume and theme

View/LoadAssets

Uses

Pygame

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
load_floor	—	pygame.image	FileNotFoundError
load_background	—	pygame.image	FileNotFoundError
load_character	—	seq of pygame.image	FileNotFoundError
load_all_obstacle	—	seq of pygame.image	FileNotFoundError
load_all_powerups	—	seq of pygame.image	FileNotFoundError
load_main_menu	—	pygame.image	FileNotFoundError
load_sound	—	pygame.mixer	FileNotFoundError

Semantics

State Variables

None

State Invariant

None

Assumptions

All the files are in the appropriate directory with proper names and format.

Access Routine Semantics

load_floor():

- output: $out :=$ returns a Pygame image object with the floor image loaded in.
- exception: $exc := (FLOOR_IMG \equiv \neg FileExists) \Rightarrow FileNotFoundError$

load_background():

- output: *out* := returns a Pygame image object with the background image loaded in.
- exception: *exc* := (BACKGROUND_IMG $\equiv \neg FileExists$) $\Rightarrow FileNotFoundError$
load_character():
- output: *out* := returns a sequence of Pygame image objects with images of all obstacles loaded in.
- exception: *exc* := (CHARACTER_IMG $\equiv \neg FileExists$) $\Rightarrow FileNotFoundError$
load_all_obstacle():
- output: *out* := returns a Pygame image object with the floor image loaded in.
- exception: *exc* := (OBSTACLE_IMG $\equiv \neg FileExists$) $\Rightarrow FileNotFoundError$
load_all_powerups():
- output: *out* := returns a Pygame image object with the all kinds of power ups loaded in.
- exception: *exc* := (POWERUP_IMG $\equiv \neg FileExists$) $\Rightarrow FileNotFoundError$
load_main_menu():
- output: *out* := returns a Pygame image object with the main menu image loaded in.
- exception: *exc* := (MAINMENU_IMG $\equiv \neg FileExists$) $\Rightarrow FileNotFoundError$
load_sound():
- output: *out* := returns Pygame sound objects with the background music and sound effects loaded in.
- exception: *exc* := (SOUND_MP3 $\equiv \neg FileExists$) $\Rightarrow FileNotFoundError$

Local Constants

```

FLOOR_IMG = 'floor.png'
BACKGROUND_IMG = 'background.png'
CHARACTER_IMG = ['character.png', 'character_invisible.png']
OBSTACLE_IMG = ['obstacle1.png', 'obstacle2.png']
POWERUP_IMG = ['powerup1.png', 'powerup2.png', 'powerup3.png', 'powerup4.png']
MAINMENU_IMG = 'mainmenu.png'
SOUND_MP3 = ['sound1.mp3', 'sound2.mp3', 'sound3.mp3', 'sound4.mp3', 'sound5.mp3']

```

Controller/GameController

Uses

Character
Obstacle
Powerups
DetectCollision
UpdateEnvironment
Score
DisplayObstacle
DisplayPowerups
DisplayEnvironment
DisplayWindow
DisplayCharacter
PlaySound
DisplayMenu
LoadAssets

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
GameController	—	GameController	—
game_loop	—	—	—
check_user_input	—	—	—

Semantics

State Variables

game_screen: pygame.display
obstacle_list: seq of Obstacle
character : Character
powerup_list : seq of Powerups
background_music: pygame.mixer
sound_effects: pygame.mixer
floor: pygame.image
floor_position: \mathbb{N}

background: pygame.image
score_count: Score

State Invariant

None

Assumptions

None

Access Routine Semantics

GameController():

- transition: game_screen := DisplayWindow.get_game_screen()
obstacle_list, powerup_list := [], []
character := Character(game_screen, LoadAssets.load_character()[0])
background, sound_effects := LoadAssets.load_sound()
floor := LoadAssets.load_floor()
floor_position := 0
background := LoadAsset.load_background()
score_count := Score()

- output: *out* := *self*

game_loop():

- transitions: This is the main game loop of the game that will continuously run until the game current session has ended either through a collision detection or exiting the game from the pause menu. The floor image will have its position constantly moving and being updated. The events of each keyboard input will be constantly monitored to ensure the user input is registered. The score will be incremented and depending on the current score, the background color will change. There will be random types of obstacles and powerups generating at random times. There will be a constant check for collision detection between the character and obstacle, and character and powerups. If a collision between a character and obstacle has occurred, the current game state stops and continues to the exit menu to compare current score and highest score achieved.

check_user_input():

- transition: checks for user_input and calls the corresponding method. The method is responsible for handling input for character control and moving the character based on the input. The method also handles the user inputs for starting the game, quitting the game, pausing the game, and changing the settings.

Input Key	Behaviour
pygame.QUIT	System exit
pygame.KEYDOWN	
pygame.K_DOWN	character.duck() & play_duck_sound()
pygame.K_UP	character.jump() & play_jump_sound()
pygame.K_Up twice	character.double_jump() & play_jump_sound()
pygame.K_P	MenuController.pause_menu(game_screen)
pygame.KEYUP	
pygame.K_DOWN	character.stand()

Controller/MenuController

Uses

MainMenu

Syntax

Exported Access Programs

Routine name	In	Out	Exceptions
setting_menu	pygame.display	—	—
pause_menu	pygame.display	—	—
exit_menu	pygame.display	—	—

Semantics

State Variables

None

State Invariant

None

Assumptions

None

Access Routine Semantics

setting_menu(screen):

- transition: display the setting menu onto the screen and handle the volume and theme change according to user input. When the user press ‘Back’, the method terminates and control is passed back to the game controller.

pause_menu(screen):

- transition: display the pause menu onto the screen, freezing the current game state. The score count is kept constant and all the elements of the game (obstacles, character, and background) stop moving. If the ‘Resume’ button is pressed, the game data current state is unfrozen, the method terminates and the control is shifted back

to the game controller. If the 'Exit' button is pressed, the method terminates and the control is shifted back to the game controller (not saving the current game state upon exiting).

`exit_menu(screen):`

- transition: display the exit menu onto the screen and restart or quit the game according to user input. When the user presses 'Exit', the game goes back to main menu. If 'Restart' is pressed, a new game starts. the method terminates and control is passed back to the game controller whatever button is pressed.