Poppy

(Game)

Submitted by :

Salvador , Jefferson B.

Cortez , Jan Paolo

Condeza , Ancess L.

Fabiosa , Julie-Ann

BSIT – 2A

Submitted to :

Ms. Elanie Vizconde

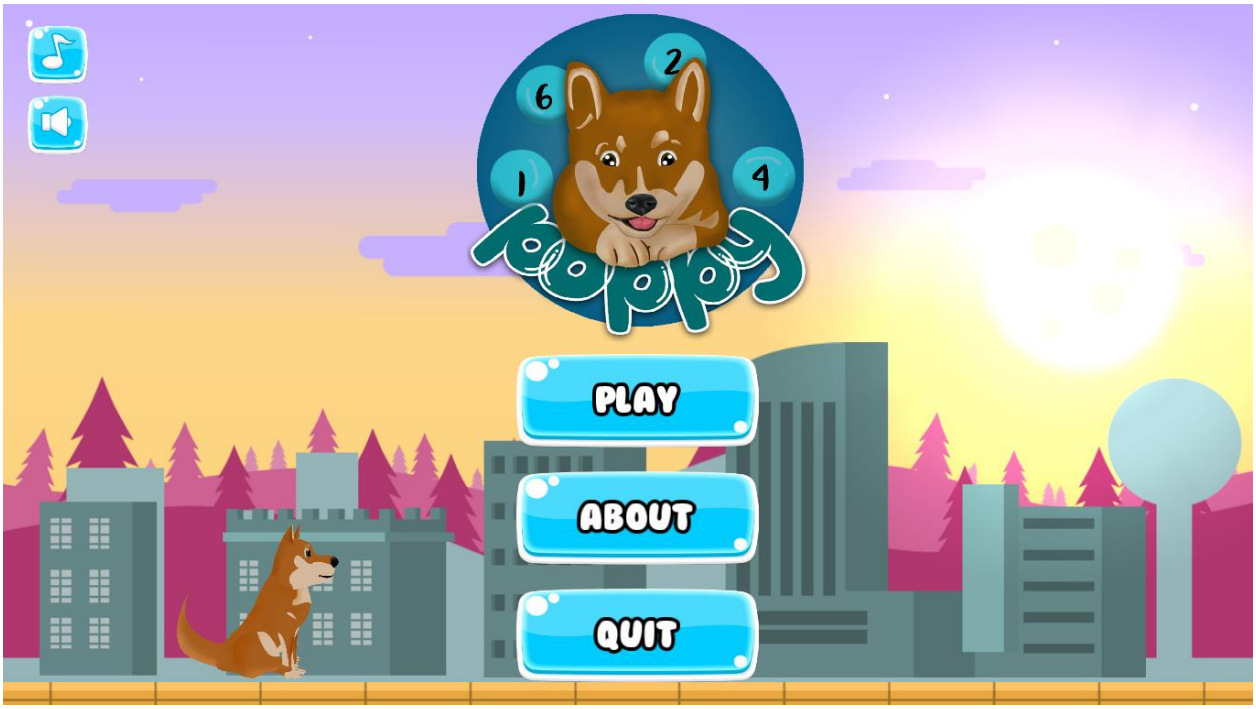
ACKNOWLEDGEMENT

We would like to mention our sincere gratitude towards our professor in this subject (Data Structure and Algorithm), Ms. Elanie Vizconde, for giving us the opportunity to experience this kind of project. To our family, who understand us for every overnight we spent just to do this project. And especially to our Almighty God, who gives us strength and intelligence.

TABLE OF CONTENTS

CODE/FUNCTION EXPLANATION

MAIN MENU

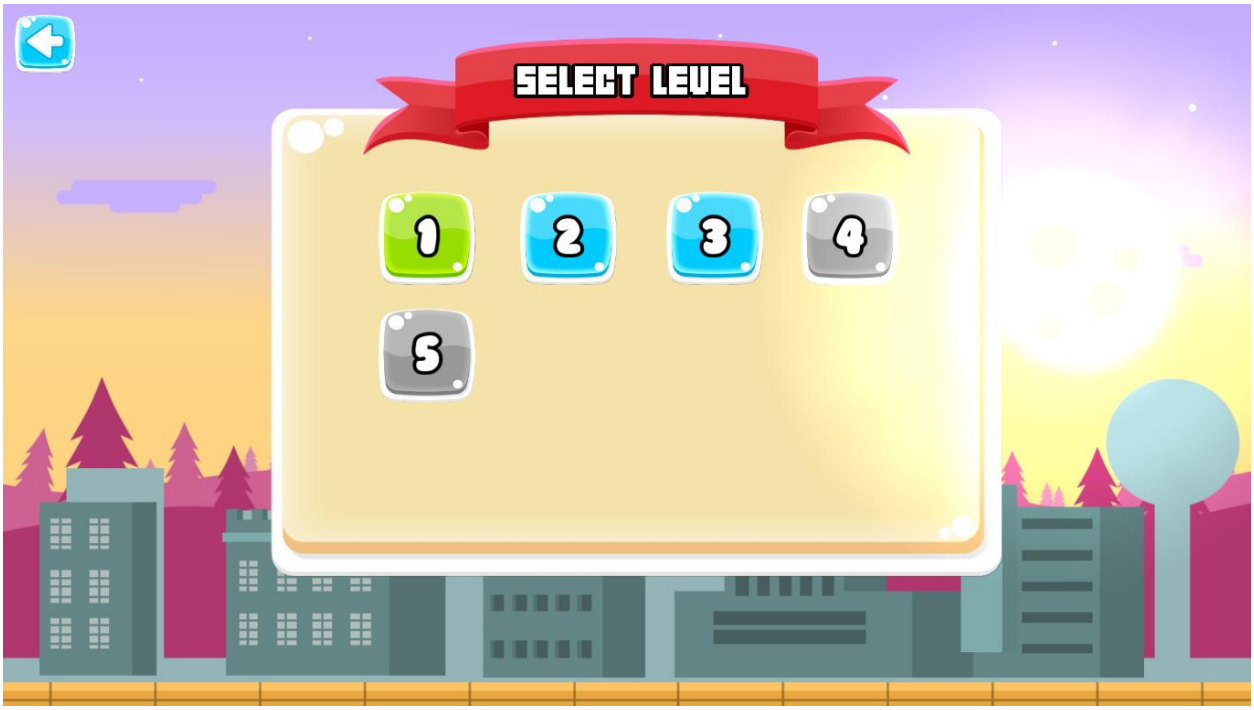


//Codes

```
extends Control


#called at the start when opening the game
#preloads the level select screen to make the transition between main menu and level seect faster
func _ready():
	preload("res://Scenes/Levels/LevelSelect.tscn")


#called when the play button is clicked
#go to the level selection screen

func _on_play_btn_pressed():
	$btn_click.play()
	$Fade.fadeIn()
	yield($Fade, "fade_end")
	get_tree().change_scene("res://Scenes/Levels/LevelSelect.tscn")


#called when the about button is clicked
#go to the about screen
func _on_about_btn_pressed():
	$btn_click.play()
	$Fade.fadeIn()
	yield($Fade, "fade_end")
	get_tree().change_scene("res://Scenes/Levels/About.tscn")


#called when the music button is toggled
#set the game music on/off
func _on_music_btn_toggled(button_pressed):
	$btn_click.play()
	AudioServer.set_bus_mute(AudioServer.get_bus_index("Music"), button_pressed)


#called when the sound button is toggled
#set the game sound on/off
func _on_sound_btn_toggled(button_pressed):
	$btn_click.play()
	AudioServer.set_bus_mute(AudioServer.get_bus_index("Sound"), button_pressed)
```

CODE/FUNCTION EXPLANATION

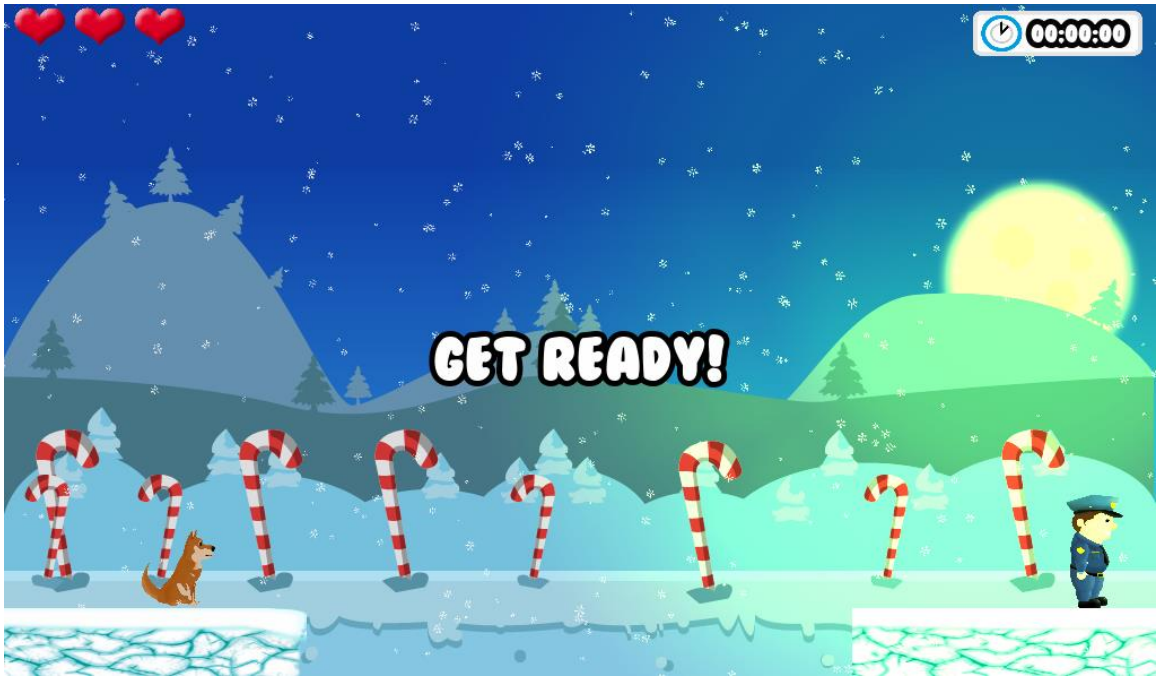
GAME LEVELS

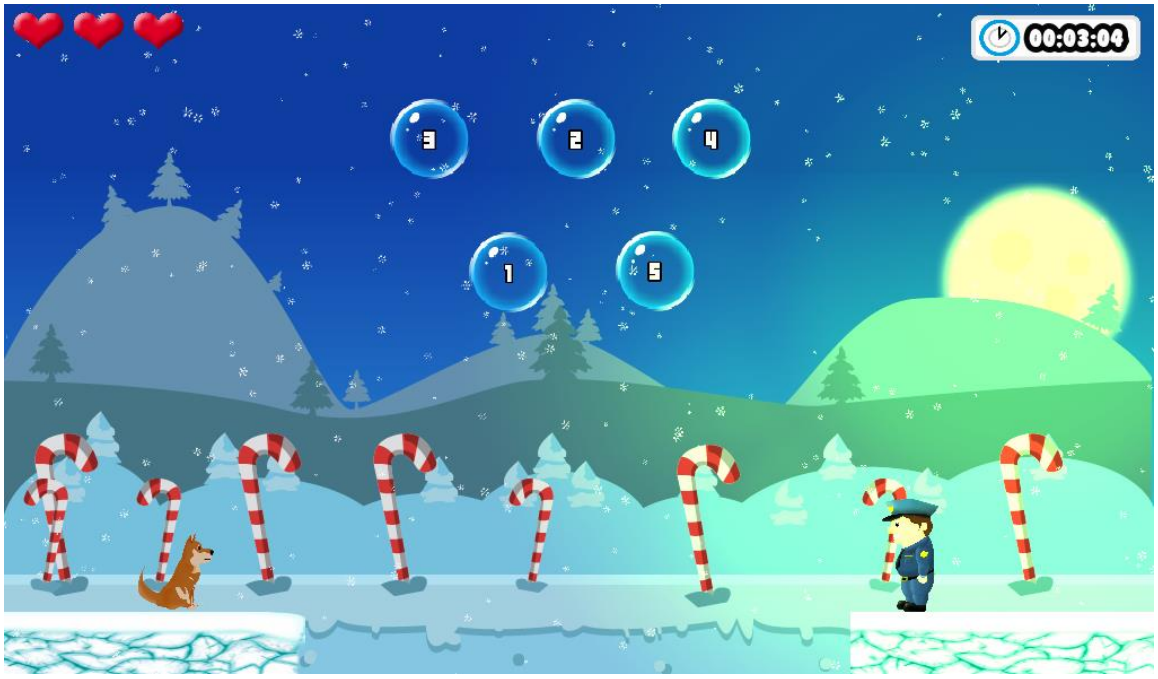


//Codes

```
extends Control

var level_to_load

var show_poster = true

#called when the level selection loads

#show poppy's missing poster if show_poster is true

func _ready():

	if show_poster == true:


	yield(get_tree().create_timer(1), "timeout")

		$poster_obj.show()


	$poster_obj/paper_sound.play()

#called when the player clicked anywhere on the screen to hide the poster if it is shown

func _input(event):

	if show_poster == true and $poster_obj.visible == true:

		if event is InputEventMouseButton && event.button_index == BUTTON_LEFT && event.is_pressed():


	$poster_obj/paper_sound.play()

			$poster_obj.hide()

#called when one of the level buttons is clicked

#store the level data sent by the button clicked to level_to_load variable

#show the level info window

func _on_level_btn_send_level_data(level, level_data):

	level_to_load = level_data

	$level_info.showLevelInfo(level)

	print(level_data)

#called when the close button on level info window is clicked

#closes the level info window

func _on_close_pressed():

	$btn_click.play()

	$level_info.hide()


#called when the play button on the level info window is clicked

#load the level selected by using the level data sent by the level button in the level_to_load variable

func _on_play_pressed():

	$btn_click.play()

	$Fade.fadeIn()

	yield($Fade, "fade_end")

	get_tree().change_scene(level_to_load)


#called when the back button is clicked

#go back to the main menu

func _on_back_btn_pressed():

	$btn_click.play()

	$Fade.fadeIn()

	yield($Fade, "fade_end")

	get_tree().change_scene("res://Scenes/Levels/MainMenu.tscn
```

GAME LEVEL INTERFACE





//Codes

```
extends Node


export (String) var char_name
export (String) var dialog
export (Texture) var char_frame
export (NodePath) var character_node_path


export (String) var current_level_dir
export (String) var next_level_dir
export (int) var multiplier = 1


var bubbles = []
var bridges = []
var arr = []
var lives = 3
var ctr = 0
var idx = 0
var x
var win = false
var score = 0
var character


#called when the level started
#-store all bubbles in the "bubbles" array
#-call initializeBubble() function to put random numbers in every bubble then store the numbes in a seperate array
#-sort the array where the numbes are stored
#-hide bubbles when the level starts
#-show "Get Ready!"
#-after few seconds unhide bubbles then the level will start
func _ready():
	character = get_node(character_node_path)
	randomize()
	$HUD.current_level = current_level_dir
	$HUD.next_level = next_level_dir
	bubbles = get_tree().get_nodes_in_group("bubbles")
	initializeBubbles()
	arr.sort()
	get_tree().call_group("bubbles", "hide")

	$HUD.show_message("Get Ready!")
	yield($HUD/messageTimer, "timeout")
	$HUD.timerStart()
	get_tree().call_group("bubbles", "show")

func _process(delta):
	pass


#called when the player won
#-stop the timer
```

```
#-calculate score based on the time

#-build the brdige for the dog to cross

#-after 4 seconds the do will cross the bridge

func win():

	win = true

	$HUD.timerStop()


	if lives == 3 and ($HUD/Timer.m2 < 1 and $HUD/Timer.m1 == 0):

		score = 3

	elif lives == 2 or (($HUD/Timer.m2 >= 1 and $HUD/Timer.m2 < 2) and $HUD/Timer.m1 == 0):

		score = 2

	elif lives == 1 or ($HUD/Timer.m2 >= 2 or $HUD/Timer.m1 != 0):

		score = 1


	bridges = get_tree().get_nodes_in_group("bridge")

	buildBridge()

	yield(get_tree().create_timer(4), "timeout")

	$Poppy.crossBridge()


#called when the player loses

#-stop the timer

#-dog will jump then it's game over

func game_over():

	win = false

	$HUD.timerStop()

	$background.mouse_filter = Control.MOUSE_FILTER_STOP

	$Poppy.crossBridge()


#used to put random numbers ong the bubbles

#-traverse each bubble in the "bubbles" array

#-for each bubble generate a random number then compare if the number generated is already in the "arr" array if not then put the number on the bubble and store in "arr" array

#-"arr" is an array where random numbers on bubbles are stored then it will be sorted later to use for comparison when the user pops a bubble

func initializeBubbles():

	while ctr < bubbles.size():

		x  = randi() % (bubbles.size() * multiplier) + 1

		if not (x in arr):

			bubbles[ctr].x = x

			bubbles[ctr].set_num()

			arr.append(x)

			ctr += 1


#called when the user popped a bubble

#-when a bubble is popped the number on the bubble will be compared to the number in "arr"

#-suppose that the numbers on the bubbles are [3, 4, 5, 1, 2] those numbers will be stored in the "arr" array then sorted, arr = [1, 2, 3, 4, 5] now

#-for example if the user pops a bubble with number 5 it will be compared to the first index of the sorted "arr" array which in this case is equals to 1, it is not the same so the bubble will
```

be hidden for a few seconds then will be shown again after

#-if the user pops a bubble with number number 1 t will be compared to the first index which is still one, it is the same so the bubble will be popped permanently and the comparison will be moved to the next index

#-the process will just repeat for each right or wrong pop

```
func _on_Bubble_pressed(obj):
	if obj.x == arr[idx]:
		obj.pop()
		idx += 1
	else:
		obj.hide_bubble()
		$HUD.removeLife()
		lives -= 1

	if idx >= arr.size():
		win()
	if lives <= 0:
		game_over()
```

#used to build the bridge that the the dog will cross

```
func buildBridge():
	for bridge in bridges:

	yield(get_tree().create_timer(0.5), "timeout")
		bridge.show()
		bridge.allocate()
```

#used to detect if the dog reached the other end of the bridge

#-if the dog reached the other end "win" window will be shown

```
func _on_Poppy_stopped():
	$HUD.showDialogBox(char_name, dialog, score, char_frame)
	character.stop()
```

#used to detect if the dog fell

#-if the dog fell "game over" window will be shown

```
func _on_Poppy_fall():
	score = 0
	$HUD.showLose()
```

CODE/FUNCTION EXPLANATION

//Codes

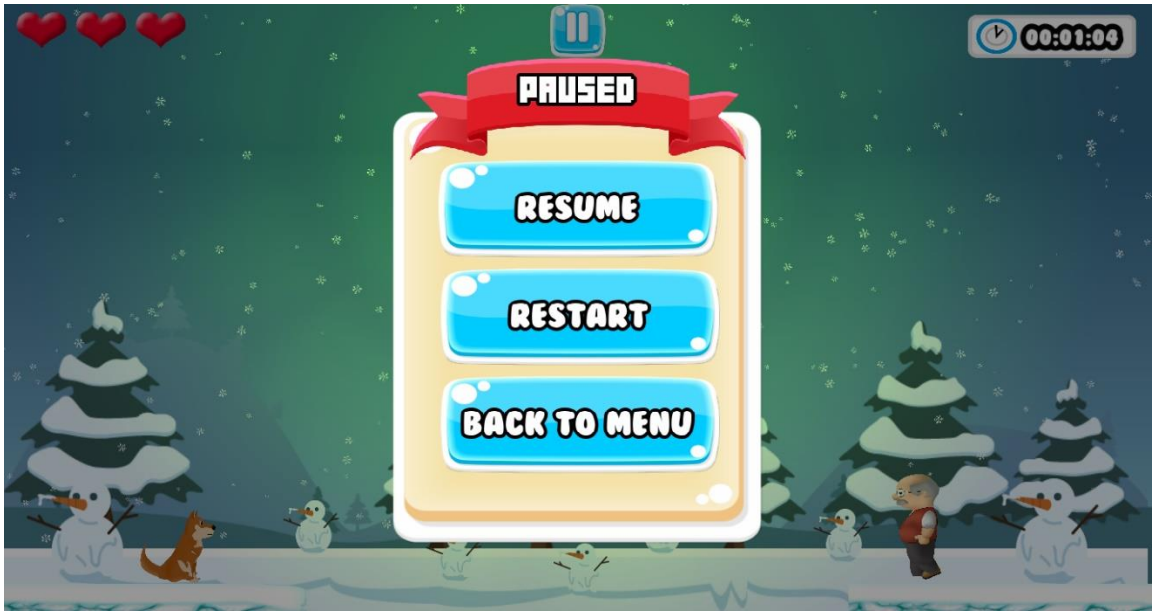
```
extends Node


export (String) var char_name
export (String) var dialog
export (Texture) var char_frame
export (NodePath) var character_node_path
export (String) var current_level_dir
export (String) var next_level_dir
export (int) var multiplier = 1


var bubbles = []
var bridges = []
var arr = []
var lives = 3
var ctr = 0
var idx = 0
var x
var win = false
var score = 0
var character


#called when the level starts
#calls initializeBubbles() to put random numbers inside every bubbles in the level
#when the bubbles was initilized start the game
func _ready():
	character = get_node(character_node_path)
	randomize()
	$HUD.current_level = current_level_dir
	$HUD.next_level = next_level_dir
	bubbles = get_tree().get_nodes_in_group("bubbles")
	initializeBubbles()
	arr.sort()
	get_tree().call_group("bubbles", "hide")


	$HUD.show_message("Get Ready!")
	yield($HUD/messageTimer, "timeout")
	$HUD.timerStart()
	get_tree().call_group("bubbles", "show")


#called every frame to watch out if the key for "pause" or "level select" is pressed
func _process(delta):
	if Input.is_action_just_pressed("level_select"):
		$HUD/Fade.fadeIn()
		yield($HUD/Fade, "fade_end")


	get_tree().change_scene("res://Scenes/Levels/LevelSelect.tscn")
	if Input.is_action_just_pressed("pause"):
		pause()
```

```
#called when the player wins

#stop the timer

#calculate the score

#show/build the bridge

#once the bridge is completed, make poppy
cross the bridge

func win():

	win = true

	$HUD.timerStop()



	if lives == 3 and ($HUD/Timer.m2 < 1
and $HUD/Timer.m1 == 0):

		score = 3

	elif lives == 2 or (($HUD/Timer.m2 >= 1
and $HUD/Timer.m2 < 2) and $HUD/Timer.m1
== 0):

		score = 2

	elif lives == 1 or ($HUD/Timer.m2 >= 2
or $HUD/Timer.m1 != 0):

		score = 1



	bridges =
get_tree().get_nodes_in_group("bridge")

	buildBridge()

	yield(get_tree().create_timer(4),
"timeout")

	$Poppy.crossBridge()


#called when the player loses

#stop the game timer

func game_over():

	win = false

	$HUD.timerStop()

	$background.mouse_filter =
Control.MOUSE_FILTER_STOP

	$Poppy.crossBridge()


#called from the _ready() function

#generate random number -> check if the
number is already in the array that will be used
later for comparison when the user pops a
bubble -> if already exist, generate another
number -> if not, put the number inside the
bubble then store it also in the comparison array

#repeat until all bubbles has a number

func initializeBubbles():

	while ctr < bubbles.size():

		x  = randi() % (bubbles.size() *
multiplier) + 1

		if not (x in arr):

			bubbles[ctr].x = x

			bubbles[ctr].set_num()

			arr.append(x)

			ctr += 1


#called when the player pops a bubble

#check if the player pops the right bubble:

#-if right, move to the next index of comparison
array

#-if wrong, remove 1 life

#if the user pops the last bubble then call win()
to tell the player that he/she won
```

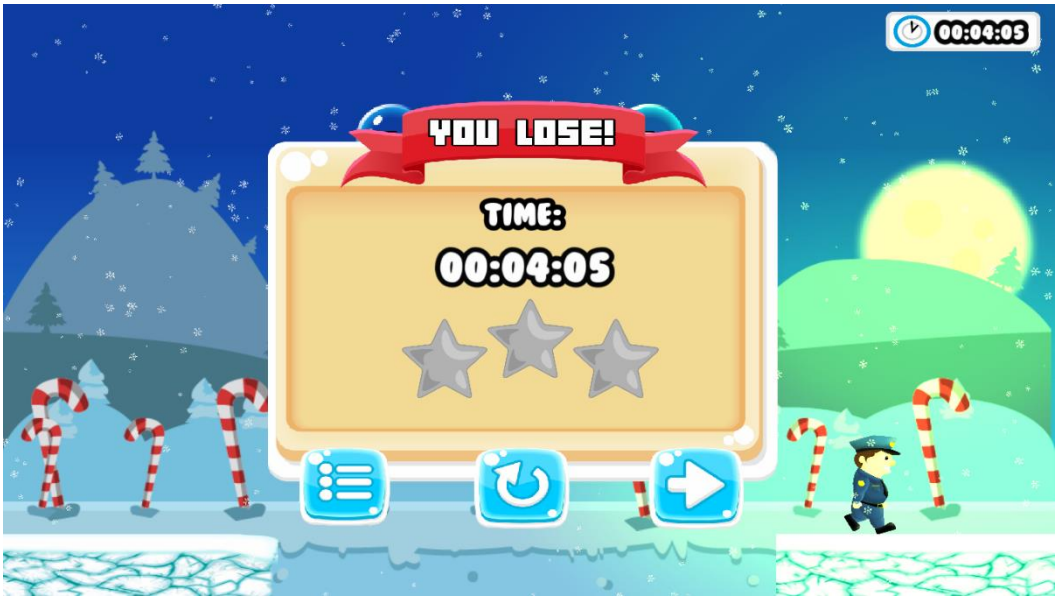
```
#if the user pops a wrong bubble and only has 1 life left, remove the last life then call gameover() function to tell that he/she lose
func _on_Bubble_pressed(obj):
	if obj.x == arr[idx]:
		obj.pop()
		idx += 1
	else:
		obj.hide_bubble()
		$HUD.removeLife()
		lives -= 1

	if idx >= arr.size():
		win()
	if lives <= 0:
		game_over()

#called in the win() function to contruct the bridge for poppy to cross when the player won
func buildBridge():
	for bridge in bridges:

	yield(get_tree().create_timer(0.5), "timeout")
		bridge.show()
		bridge.allocate()

#called if the player pressed "P" to pause/resume the game
#hide all the bubbles and show "paused" message when the game is paused
#show the bubbles again on resume
func pause():
	if get_tree().paused == false:

	get_tree().call_group("bubbles", "hide")
	elif get_tree().paused == true:

	get_tree().call_group("bubbles", "show")

	$HUD/Dim.visible = not $HUD/Dim.visible
	$HUD/PauseLabel.visible = not $HUD/PauseLabel.visible
	get_tree().paused = not get_tree().paused

#called when poppy reached the other end of the bridge
#shows the conversation(dialog box) between poppy and the character on each level after reaching the end of the bridge
#poppy reaching the other end of the bridge indicates that the player won
func _on_Poppy_stopped():
	$HUD.showDialogBox(char_name, dialog, score, char_frame)
	character.stop()

#called when poppy fell when he tries to cross(jump) without the bridge
#set the score to 0
#show lose window
```

```
func _on_Poppy_fall():
	score = 0
	$poppy_fall.play()
	yield($poppy_fall, "finished")
	$HUD.showLose()
```

//when the game paused



//when you lose in the game

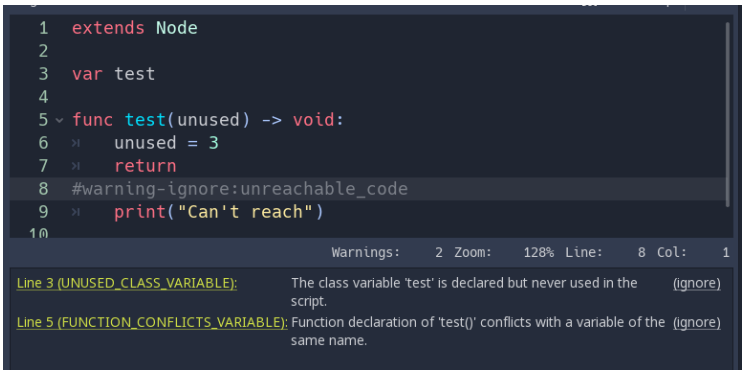


LANGUAGE OR SOFTWARE USED

For this game project, the software we used is Godot. Godot is a free and open-source game engine released under the MIT license. It was initially developed for several companies in Latin America before its public release. The development environment runs on Windows, macOS, and Linux. It can create games for PCs, mobile devices and the Web platform.

What coding language does Godot use?

Scripting. Godot games are created either in C++, C#, languages with GDNative bindings such as Rust, Nim, D, or by using its own scripting language, GDScript, a high level, dynamically typed programming language very similar to Python.



LANGUAGE OR SOFTWARE USED

Rendering

The graphics engine uses OpenGL ES 3.0 for all supported platforms; otherwise, OpenGL ES 2.0 is used. Future support for Vulkan is also planned. It supports normal mapping, specularity, dynamic shadows using shadow maps and full-screen post-processing effects like FXAA, bloom, DOF, HDR, and gamma correction. A simplified shader language similar to GLSL is also incorporated; shaders can be used for materials and post-processing. Alternatively, they can be created by manipulating nodes in a visual editor.

There is also a separate 2D graphics engine, which can operate independently of the 3D one. Examples of 2D engine features include lights, shadows, shaders, tile sets, parallax scrolling, polygons, animations, physics and particles. It is also possible to mix 2D and 3D using a 'viewport node'.

Other features

Godot contains an animation system with a GUI for editing skeletal animation, blending, animation trees, morphing and realtime cutscenes. Almost any variable defined or created-on-a-game entity can be animated. The engine uses Bullet for 3D physics simulation.

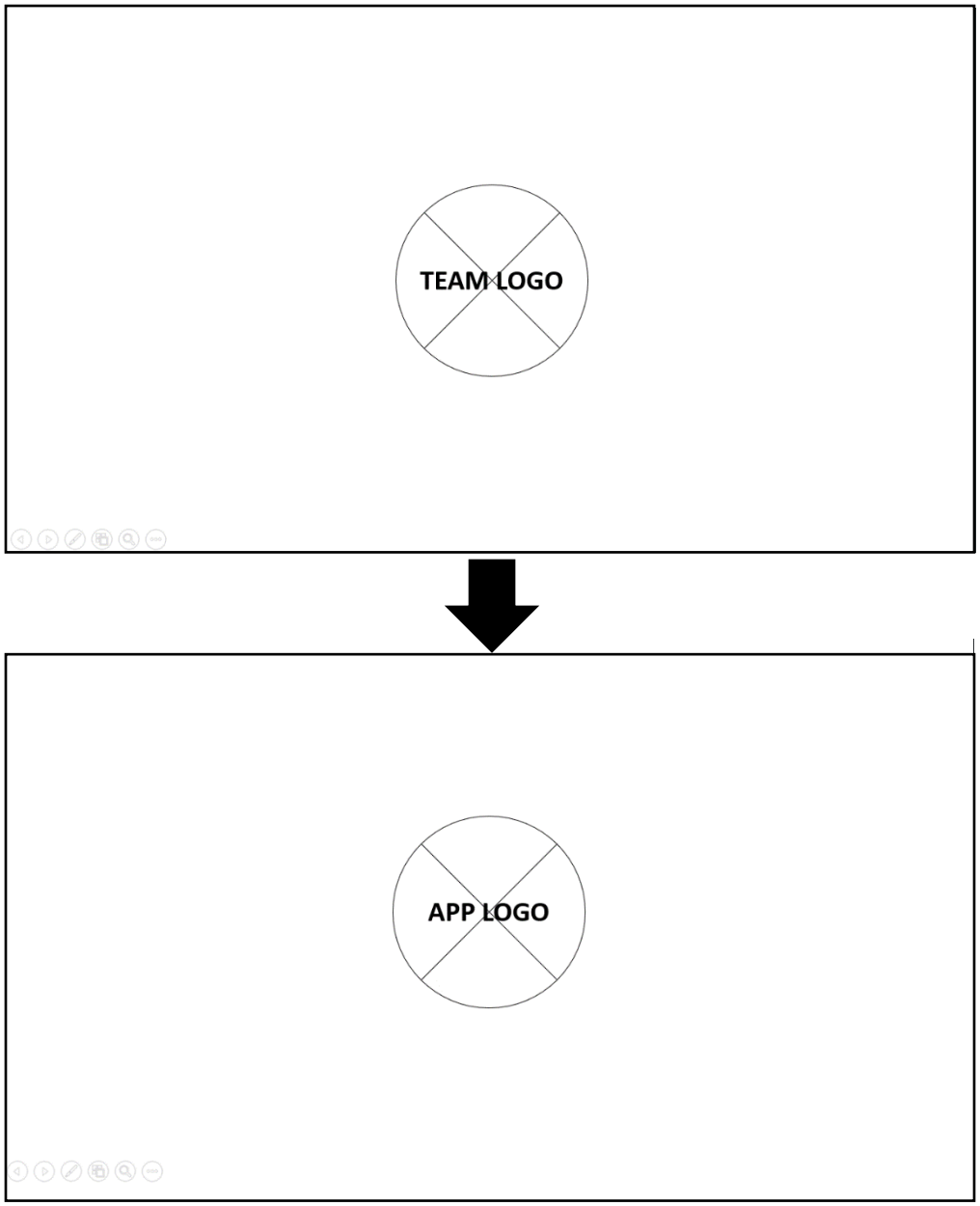
LANGUAGE OR SOFTWARE USED

Also, we used Adobe Photoshop CS6 , Adobe Photoshop is a raster graphics editor developed and published by Adobe Inc. for Windows and macOS. We use this for our animation and graphics in the game.

WIRE FRAME

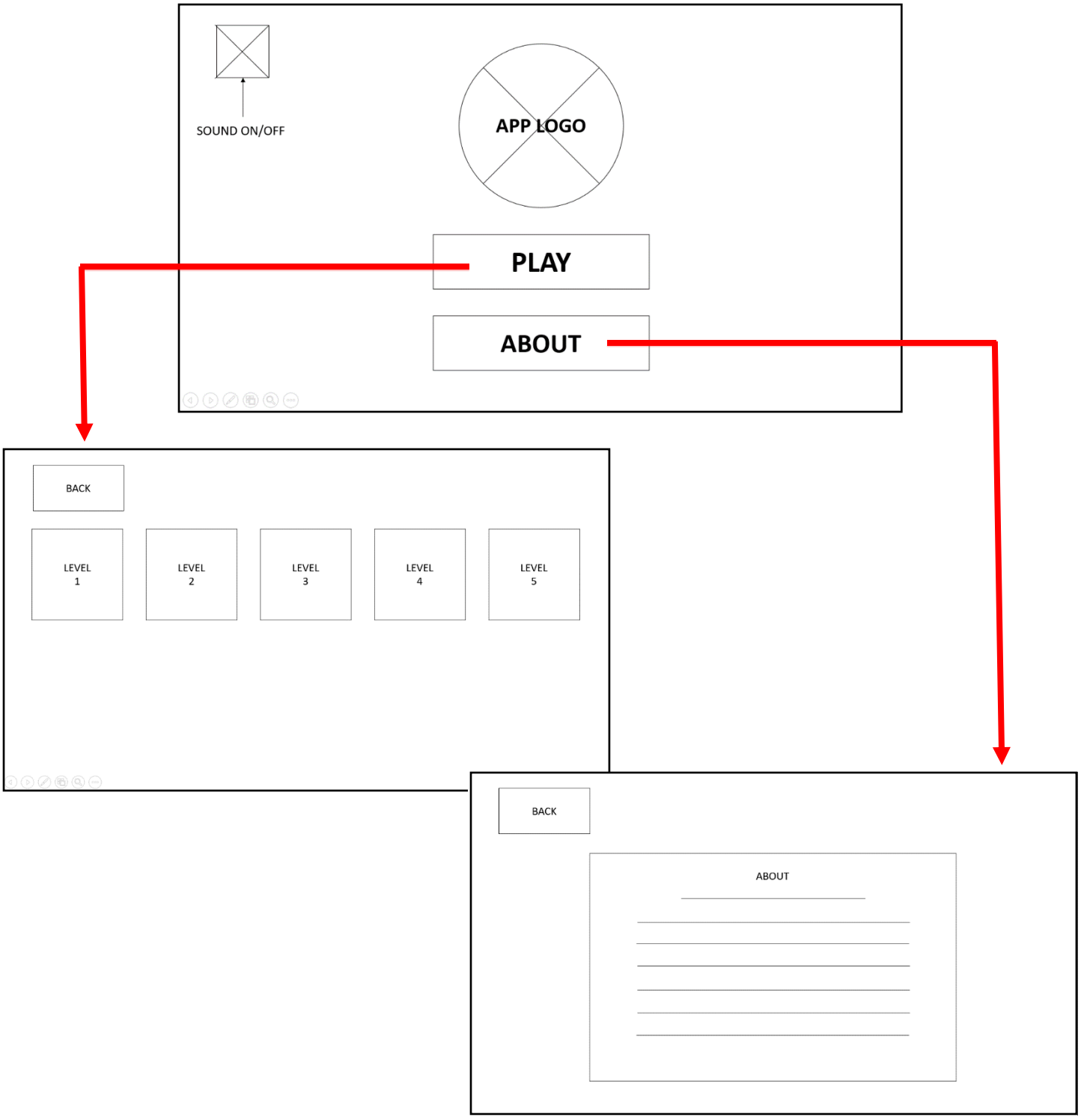
Wire Frames are a **visual representation** of an interface; used to communicate the **structure**, **content**, **information hierarchy**, **functionality**, and **behavior** of an interface.

Starting page

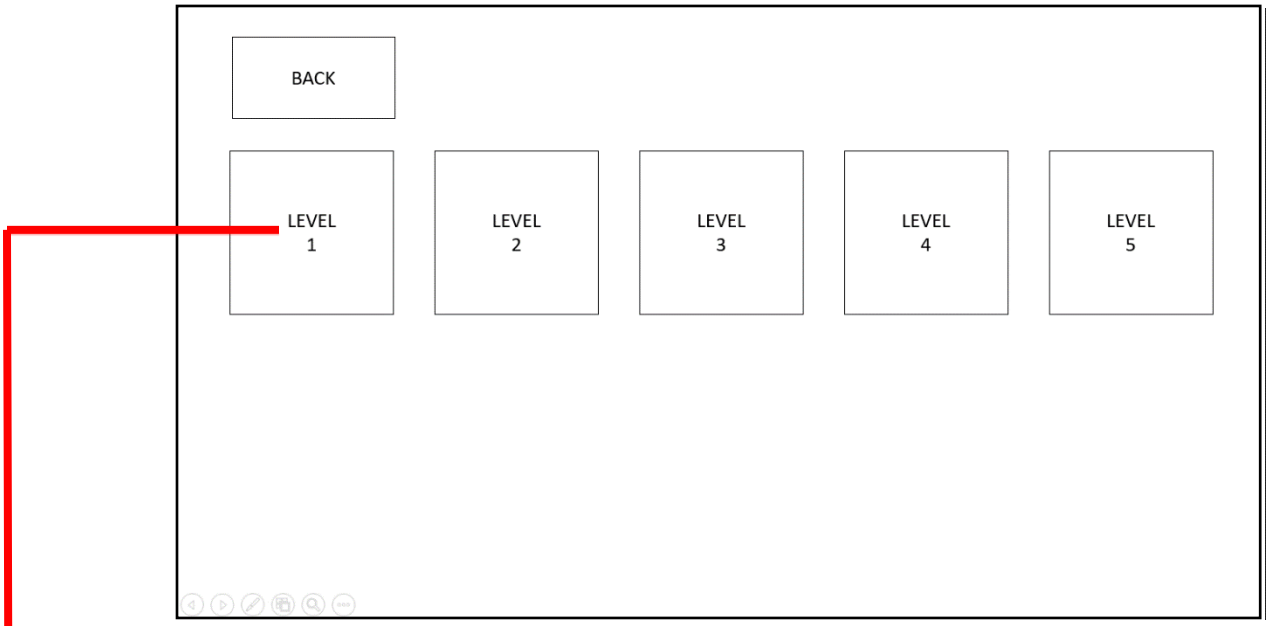


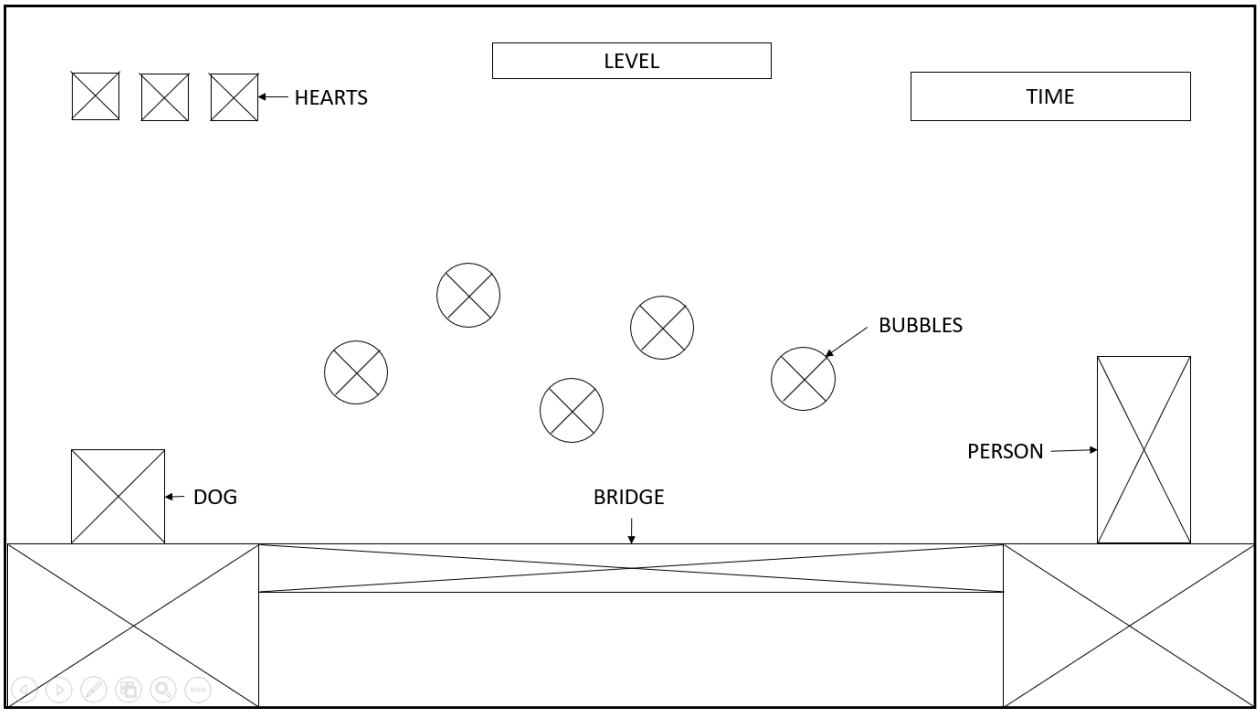
WIRE FRAME

Home Page/Menu

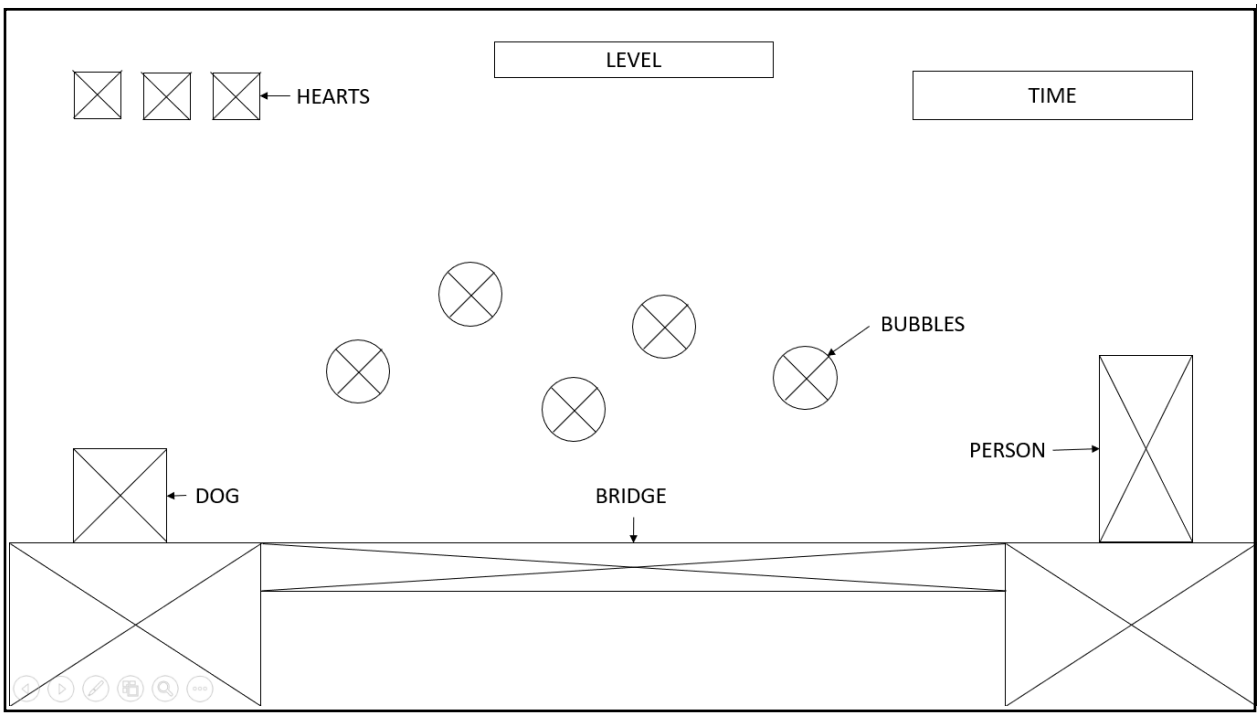


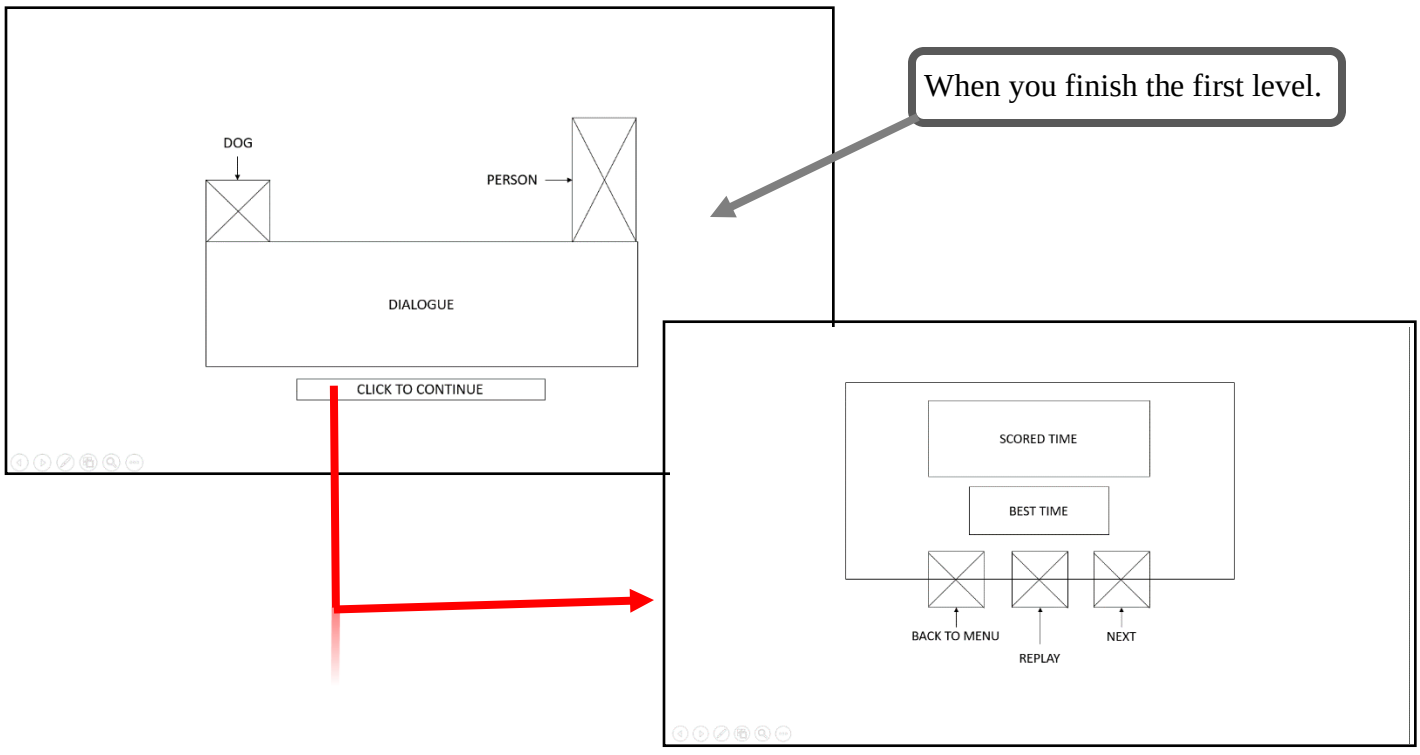
Game Levels and the Game Interface



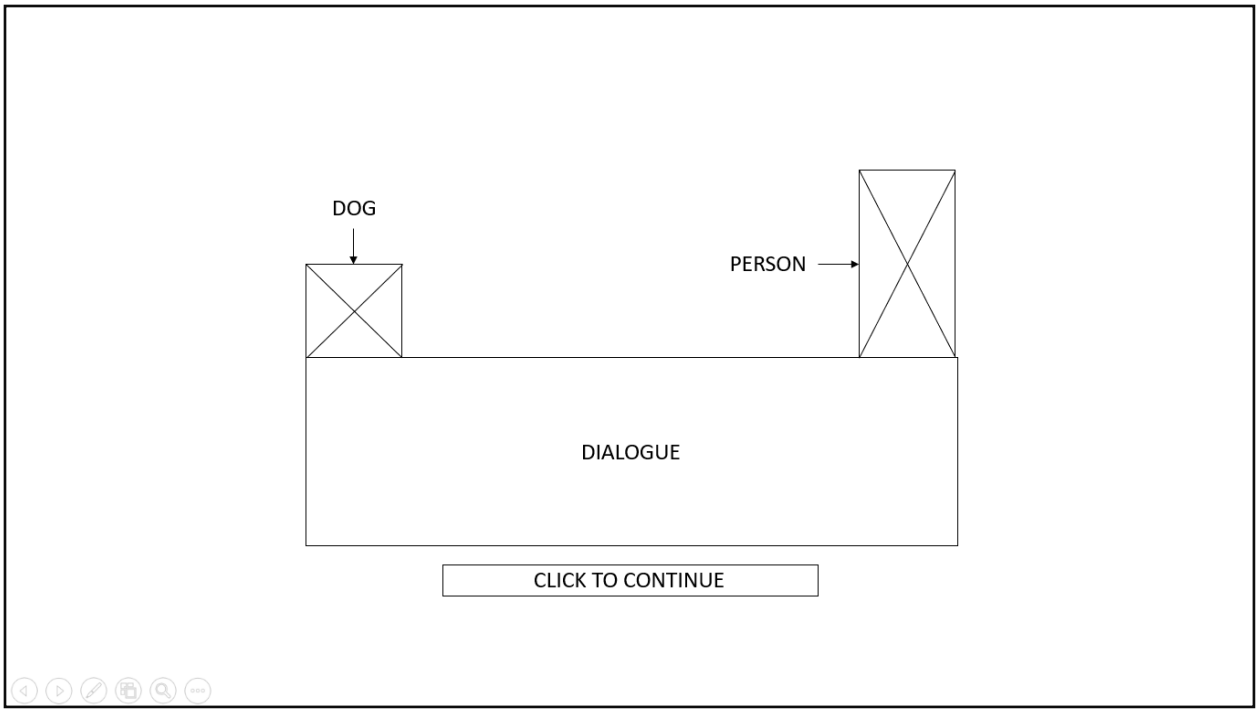


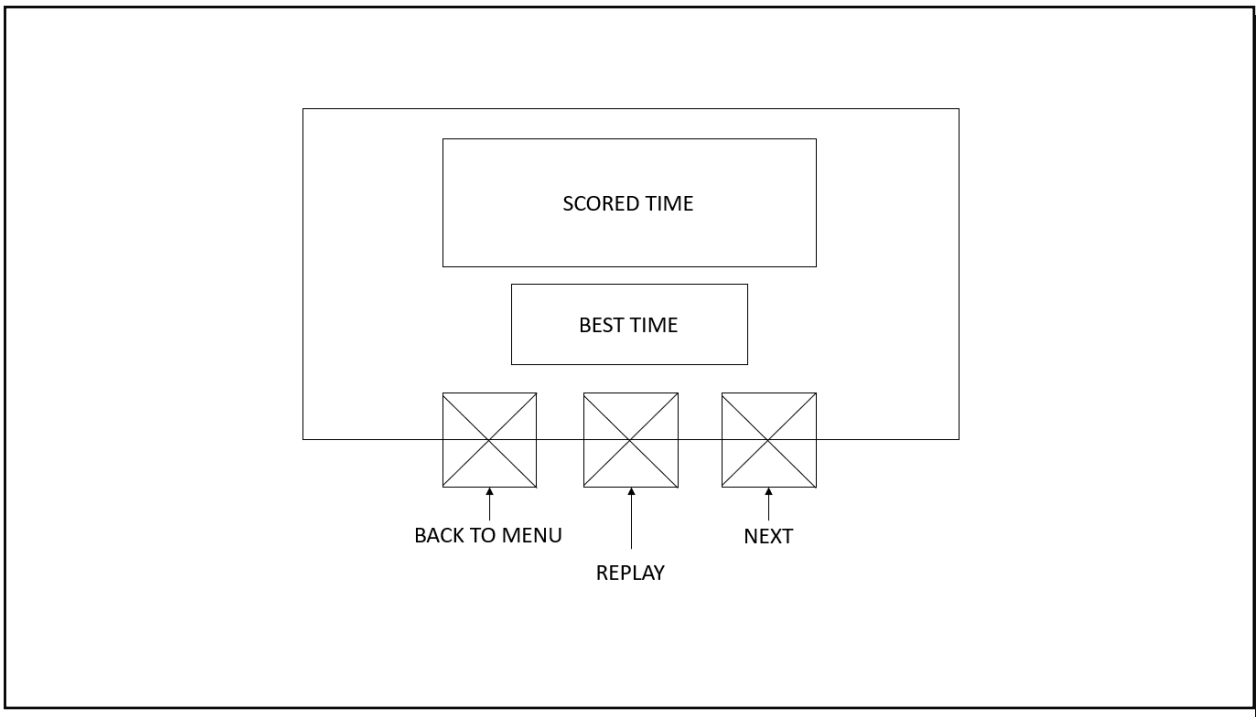
WIRE FRAME


LEVEL
HEARTS
TIME
BUBBLES
PERSON
DOG
BRIDGE


When you finish the first level.
DOG
PERSON
DIALOGUE
CLICK TO CONTINUE
SCORED TIME
BEST TIME
BACK TO MENU
REPLAY
NEXT

WIRE FRAME


DOG
PERSON
DIALOGUE
CLICK TO CONTINUE


SCORED TIME
BEST TIME
BACK TO MENU
REPLAY
NEXT

WIRE FRAME

As a starting up page, our **Team Logo** will flash in the screen first then the **App Logo**. It will lead you to our Home Page/Menu that includes the ff:

i. Setting for **Sound (On/Off) –** with this, you can turn on/off the sound of the game.

ii. **Play** button – The Play button is where the game will start so it will redirect you **Game Levels Page,** where there are also a **Back** button so you can go back to Menu Page.

   i. Game Levels Page – Different buttons with different levels, where the game will start if you click the **Levels** and will redirect you to our **Game Interface.**

   ii. Game Interface – the page where you will play the game and includes the following contents:

   - Heart – lives of the player.
   - Level – What level are you in?
   - Time – how much time you consume in playing this level.
   - Bubbles – bubbles with corresponding numbers to sort by popping it.
   - Dog – the player.
   - Person/s – the one who will help the player.
   - Bridge – is where will

iii. **About** button – The About button is the story of the game and the Credits.