Escaparade
Test Scenarios

Team Kroket

Table of contents

Introduction

This document contains test scenarios for our game. These scenarios could not be tested with regular coded tests. So this document will be a guideline of how these untestable aspects of our game are supposed to work. Also added is an oversight of classes tested with unit tests, and which classes could not be tested.

EscapeVR

Test oversight and scenarios for the VR client of the game.

Test coverage

Element	Coverage	Covered Lines	Missed Lines	Total Lines
nl.tudelft.kroket.scene.scenes	5,6 %	13	218	231
EscapeScene.java	5,6 %	13	218	231
nl.tudelft.kroket.input.interaction	0,0 %	0	131	131
nl.tudelft.kroket.net	0,0 %	0	102	102
nl.tudelft.kroket.screen	0,0 %	0	99	99
nl.tudelft.kroket.state.states	0,0 %	0	89	89
nl.tudelft.kroket	0,0 %	0	74	74
nl.tudelft.kroket.minigame.minigames	60,3 %	108	71	179
TapMinigame.java	45,6 %	36	43	79
ColorSequenceMinigame.java	54,1 %	33	28	61
GyroscopeMinigame.java	100,0 %	12	0	12
LockMinigame.java	100,0 %	13	0	13
PictureCodeMinigame.java	100,0 %	14	0	14
nl.tudelft.kroket.input	0,0 %	0	60	60
nl.tudelft.kroket.event	0,0 %	0	48	48
nl.tudelft.kroket.screen.screens	0,0 %	0	43	43
nl.tudelft.kroket.state	0,0 %	0	36	36
nl.tudelft.kroket.scene	51,7 %	31	29	60
SceneManager.java	51,2 %	22	21	43
Scene.java	52,9 %	9	8	17
nl.tudelft.kroket.event.events	0,0 %	0	23	23
nl.tudelft.kroket.audio	60,0 %	27	18	45
nl.tudelft.kroket.minigame	86,0 %	49	8	57
MinigameManager.java	84,6 %	33	6	39
Minigame.java	88,9 %	16	2	18
nl.tudelft.kroket.net.protocol	95,5 %	21	1	22
Protocol.java	0,0 %	0	1	1
CommandParser.java	100,0 %	21	0	21
nl.tudelft.kroket.log	100,0 %	29	0	29
Logger.java	100,0 %	29	0	29

The most part of the VR could not be tested because of the way JMonkey works. We were able to test most part of the logger, all classes in the minigame package, and the commandparser. The rest however, we have tried to test a little bit, but because of JMonkeys rootnode system, it was decided to make some scenarios of how the game should work.

Scenario 1: Initialise game

Setting	Oculus player starts the game and has Oculus on
Action	none
Reaction	The player sees a screen with “Hold on! waiting for other players” The head up display shows the text “trying to connect to server”
	No server running The head up display show the text “unable to connect to server”
	No server running The game continuously tries to connect. It throws an ConnectionException, which does not crash the game, and displays the following logger messages: “INFO ClientThread: Failed to connect. Retrying…” “INFO NetworkClient: Trying to connect to 127.0.0.1:1234…”
	Server running, no other players Head up display show the text “Trying to register player” and the logger message shows “INFO ClientThread: Trying to register client...”
	Server running, all players connected Screen shows “Hold on! waiting for other players” until all player are registered. If a player registered, the text “[playername] has entered the game’’ appears.

Scenario 2: Start the game

Setting	Game is connected to the server, Oculus player has seen the text “Trying to register player” and is spawned in the escape room.
Action	Game gets a start message from the server
Reaction	If two mobile players are registered, the texts disappear and the Oculus player sees the room.

Scenario 3: Start minigame A

Setting	Oculus player is near the Picasso painting
Action	Oculus player presses A on the gamepad
Reaction	The logger logging to the console should display “INFO EscapeVR: Player interacted with object painting”

	A message with BEGIN[A] should be sent to the server
	Oculus player has not initialised any other minigames The head up display shows the text “minigame A started!”
	Oculus player has initialised other minigames Nothing happens

Scenario 4: Finished minigame A

Setting	VR game has received DONE[A] message from server
Action	-
Reaction	The logger logging to the console should display “INFO PictureCodeMinigame: Minigame A completed.”
	The safe in the room should have an open door now
	The head up display shows the text “minigame A finished!”

Scenario 5: Start minigame B

Setting	Oculus player is near the desk with laptop/egg painting
Action	Oculus player presses A on the gamepad
Reaction	*Minigame A has not been finished* Nothing happens
	Minigame A has been finished (DONE[A] received from server) The head up display shows the text “minigame B started!”
	A message with BEGIN[B] is sent to the server
	The logger logging to the console should display “INFO EscapeVR: Player interacted with object x

Scenario 6: Play minigame B

Setting	Minigame B has been initialised
Action	-
Reaction	Oculus player has to press a certain sequence of A,B,X,Y buttons

Scenario 7: Finishing minigame B

Setting	VR game has received DONE[B] message from server
Action	-
Reaction	*Minigames A and B have finished (DONE[A] and DONE[B] received from server)* Left from the door, 4 colored buttons appear
	The logger logging to the console should display INFO TapMinigame: Minigame B completed.”
	The head up display shows the text “minigame B finished!”
	Minigame B failed by one or more players VR client should get a RESTART message, and minigame b can be played again.

Scenario 8: Start minigame C

Setting	Buttons have appeared, Oculus player is near the buttons
Action	Oculus player presses A on the gamepad
Reaction	*Minigames A and B have finished (*Oculus player has to enter a color code with A,B,X,Y that is given by the smartphone players.
	Minigames A and B have not been finished Nothing happens

Scenario 9: Finish minigame C

Setting	Minigame C has been initialised
Action	Oculus player has entered the right sequence of A,B,X,Y
Reaction	The head up display shows the text “minigame B finished!”
	The logger logging to the console should display INFO ColorSequenceMinigame: Minigame C completed.”

Scenario 10: Finish minigame D

Setting	Minigames A, B and C have been finished
Action	Player interacts with the door

Reaction	VR client sends BEGIN[D] message to server
	A painting with question mark appears in the room, when the player interacts with it, a number appears and a new painting with question mark. There are three paintings.
	Player has received DONE[D] message from server, within timelimit The door can open and the game is won

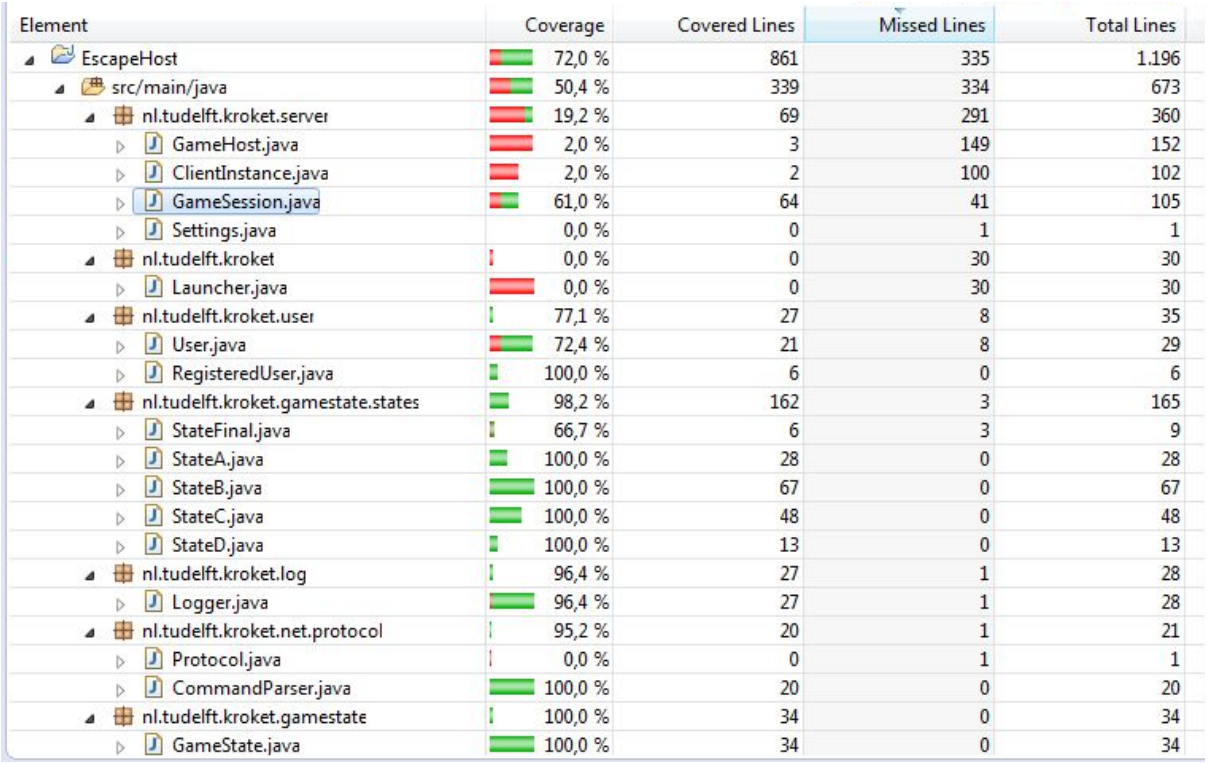
Scenario 11: Time is up

Setting	The server has sent a GAMELOST message because time is up
Action	-
Reaction	Game displays a game lost overlay

EscapeHost

Test oversight and scenarios for the host of the game.

Test coverage



Element	Coverage	Covered Lines	Missed Lines	Total Lines
EscapeHost	72,0 %	861	335	1.196
src/main/java	50,4 %	339	334	673
nl.tudelft.kroket.server	19,2 %	69	291	360
GameHost.java	2,0 %	3	149	152
ClientInstance.java	2,0 %	2	100	102
GameSession.java	61,0 %	64	41	105
Settings.java	0,0 %	0	1	1
nl.tudelft.kroket	0,0 %	0	30	30
Launcher.java	0,0 %	0	30	30
nl.tudelft.kroket.user	77,1 %	27	8	35
User.java	72,4 %	21	8	29
RegisteredUser.java	100,0 %	6	0	6
nl.tudelft.kroket.gamestate.states	98,2 %	162	3	165
StateFinal.java	66,7 %	6	3	9
StateA.java	100,0 %	28	0	28
StateB.java	100,0 %	67	0	67
StateC.java	100,0 %	48	0	48
StateD.java	100,0 %	13	0	13
nl.tudelft.kroket.log	96,4 %	27	1	28
Logger.java	96,4 %	27	1	28
nl.tudelft.kroket.net.protocol	95,2 %	20	1	21
Protocol.java	0,0 %	0	1	1
CommandParser.java	100,0 %	20	0	20
nl.tudelft.kroket.gamestate	100,0 %	34	0	34
GameState.java	100,0 %	34	0	34

As can be seen above the server package is not really tested. This is because it is the place where messages are sent, and a real connection is needed. The GameSession class could be tested partly. The Launcher only contains a main method, which we will not test since the host would not even start if this was broken. The settings and protocol contain no logic. The values are checked in a test, but obviously this does not give coverage. As for the untested parts of GameSession and User, these contain some methods for sending messages and other things that require a real connection.

Scenario 1: No players connected

Setting	Server is running, no players are connected
Action	-
Reaction	Server continuously checks for players and if the game is ready (game is ready when 2 smarthpone players and 1 oculus player are connected)
	Server shows log messages “INFO EscapeHost: Game not ready.” and “No players currently registered”

Scenario 2: Registering players

Setting	Two smartphone players and an Oculus player have started the game and connected to the host, server has gotten all REGISTER messages
Action	-
Reaction	Logger shows messages: “INFO EscapeHost: Server is ready to host game.” “INFO EscapeHost: Starting game…” “INFO EscapeHost: Game is active.”
	Host prints connected clients and registered players to console, an example: Connected clients: Client: /145.94.176.188:4495 Client: /145.94.209.12:60555 Client: /127.0.0.1:64670 Registered players: Registered: User [insert user name] - /145.94.176.188:4495 - MOBILE Registered: User [insert user name] - /145.94.209.12:60555 - MOBILE Registered: User RIFT-USER - /127.0.0.1:64670 - VIRTUAL
	Host sends START message to all clients

Scenario 3: Minigame is triggered

Setting	VR client triggers a minigame, host receives BEGIN message
Action	-
Reaction	If the minigame is allowed to be triggered - the host is in the right gamestate, e.g. the message is BEGIN[A] and the gamestate is GameState_A - the host sends the INITM message to the smartphone players. If the gamestate is not right, the host does nothing.

Scenario 4: Minigame is finished

Setting	Smartphone users have finished a minigame, host receives DONE message
Action	-
Reaction	The host sends the message to the VR client, which stops the minigame in the client

EscapeApp

Classes covered entirely by the unit tests: A_Code_Cracker_Pictureview, C_ColorSequence, CommandParser, GameOver, GameWon.

Classes covered entirely by the tests in the Test Document: Waiting_Gyroscope, Waiting_Squasher, WaitingActivity.

Classes covered by a combination unit tests and tests in the Test Document: MainActivity, A_CodeCrackerview, B_TapGame

Tests for MainActivity

Scenario's already covered by the unit tests: What happens if we try to connect without filling in a name and what happens when we fail to connect.

Scenario 1: Player connects to the server.

Setting	The player is currently in the main acitivity.
Action	The player fills in his/her name and the ip address of the host and clicks on the connect button.
Reaction	The player establishes a connection with the host. Disabling the connect button and enabling the start button. textView connectionMessage becomes "Connection established, tap START to begin!".

Scenario 2: Player receives message after connecting.

Setting	The player is currently in the main activity and has established a connection with the host.
Action	The android app receives a message START from host.
Reaction	The app will automatically go to the waiting activity.

Scenario 3: Player presses start

Setting	The player just started up the app and is currently in the main activity.
Action	The player presses the start button.
Reaction	If the player has established a connection with the server. The player successfully presses the start button and moves directly to the waiting activity.
	If the player has not established a connection with the server. Nothing happens because the start button isn't enabled till a connection has been established. So the player cannot press the start button successfully.

Tests for A_CodeCrackerCodeview

Scenario's already covered by the unit tests: what happens when we click the picture button. What happens when we enter the wrong answer.

Scenario 1: Player fills in the right answer in Activity A.

Setting	The player is currently in the A_CodeCrackerCodeview activity.
Action	The player fills in the correct solution (2719173) and presses the verify button.
Reaction	If the player has established a connection with the server. Send message DONE[A] to the host. And the mobile player goes to the waiting activity.
	If the player has not established a connection with the server. No message is send to the host. The string "ConnectionService not bound in CodeCrackerCodeView" is printed to the system and the mobile player goes to the waiting activity.

Tests for B_TapGame

Scenario's already covered by the unit tests: What happens if we click the start button once and start the timer. What happens when we click the start button twice and increase the amount.

Scenario 1: Player clicks on start

(after starting the minigame and updates the sequence text.)

Setting	The player is in B_TapGame and the timer has already started
Action	The player presses the start button.
Reaction	If the amount of clicks is equal to 25. The sequence text is updated to show the first sequence
	If the amount of clicks is equal to 50. The sequence text is updated to show the second sequence
	If the amount of clicks is equal to 75. The sequence text is updated to show the third sequence
	If the amount of clicks is equal to 100. The sequence text is updated to show the fourth sequence

Scenario 2: The timer runs out

Setting	The player is in B_TapGame and the timer has just ran out
Action	none.
Reaction	If the player has reached his/her goal of 125 clicks a message VERIFY[B] is sent to the host.
	If the player has not reached his/her goal of 125 clicks a message START[B] is sent to the host. The app then receives a message START[B] from the host and minigame B is then restarted.

Tests for Waiting_Gyroscope

Scenario 1: Sensor movement during minigame.

Setting	Minigame D_Gyroscope is active.
Action	The player moves his mobile phone up, down, left or right. In the x or y direction.

Reaction	If the mobile player moves his phone up in the y - direction. ImageView gyroimage moves up proportionally to how far the player moved his phone up in the y - direction.
	If the mobile player moves his phone down in the y - direction. ImageView gyroimage moves down proportionally to how far the player moved his phone down in the y - direction.
	If the mobile player moves his phone left in the x - direction. ImageView gyroimage moves left proportionally to how far the player moved his phone left in the x - direction.
	If the mobile player moves his phone right in the x - direction. ImageView gyroimage moves right proportionally to how far the player moved his phone right in the x - direction.

Scenario 2: Player picks up a coin.

Setting	Minigame D_Gyroscope is active.
Action	The gyro (controlled by the player) collides with a coin.
Reaction	If the coin that gyro collides with is gold the gold count is increased by three. And all coins are randomly placed on the field. The coins however can't be placed where the gyro is currently at and the coins can't be placed offscreen.
	If the coin that gyro collides with is silver the silver count is increased by two. And all coins are randomly placed on the field. The coins however can't be placed where the gyro is currently at and the coins can't be placed offscreen.
	If the coin that gyro collides with is bronze the bronze count is increased by one. And all coins are randomly placed on the field. The coins however can't be placed where the gyro is currently at and the coins can't be placed offscreen.
	If the coin that gyro collides with is the Dead coin the count is set to zero. And all coins are randomly placed on the field. The coins however can't be placed where the gyro is currently at and the coins can't be placed offscreen.

Scenario 3: Player gets score of 50.

Setting	Minigame E_squasher is active.
Action	The player's score reaches 50
Reaction	The message DONE[WAITING] is sent to the host and the coin count is set to zero again. And all coins are randomly placed on the field

Tests for Waiting_Squasher

Scenario 1: Player taps a bug.

Setting	Minigame E_squasher is active.
Action	The player taps the displayed bug.
Reaction	The squash count is increased by one. The bug image is randomized(from a set of bug images). The bugs location is randomized (ofcourse the bugs location can’t be offscreen) and the bugs rotation is randomized.

Scenario 2: Player gets score of 50.

Setting	Minigame E_squasher is active.
Action	The player gets a bug squash score of 50
Reaction	The message DONE[WAITING] is sent to the host and the squash count is set to zero. The bugs image,location and rotation are then randomized again.

Tests for WaitingActivity

Scenario 1: Player starts the gyroscope minigame.

Setting	The player is currently in the waiting activity.
Action	The player presses the start coin game button.
Reaction	The player starts and moves directly to the Waiting_Gyroscope activity.

Scenario 2: Player starts the bug squasher minigame.

Setting	The player is currently in the waiting activity.
Action	The player presses the start squasher game button.
Reaction	The player starts and moves directly to the Waiting_Squasher activity.