MT-01 — Audio Feedback on Player Interaction

Requirement(s)

- UR_AUDIO

Component / System

- `AudioSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)
- Keyboard & Mouse
- Game launched via IDE / Gradle

Preconditions

- Game launches successfully
- Audio output enabled
- Player can move freely

Test Steps

1. Launch the game from the main menu
2. Start a new game session
3. Move the player character
4. Trigger interactions that should produce sound (e.g. collecting an item, being attacked)
5. Listen for audio feedback

Expected Result

- Audio plays when interactions occur
- No noticeable delay or distortion
- Game continues without crashing

Actual Result

- Background music plays normally
- Footsteps play normally
- Sound effects play normally when attacked
- Item pickup sound effects play correctly

Test Outcome

- PASS

Notes

Audio playback relies on LibGDX runtime behaviour and hardware interaction, making automated unit testing unsuitable.

MT-02 — Rendering and Visual Stability During Gameplay

Requirement(s)

- UR_UI
- UR_THEME
- FR_THEME

Component / System

- `RenderingSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)
- Keyboard & Mouse
- Game launched via IDE / Gradle

Preconditions

- Game starts without rendering errors

- Player is visible on screen

Test Steps

1. Launch the game and start a level
2. Observe initial rendering of the maze
3. Move the player through different areas
4. Trigger animations and collisions
5. Observe rendering throughout gameplay

Expected Result

- All entities render correctly
- No flickering or missing textures
- Visuals reflect a university-themed environment

Actual Result

- Solid rendering is functioning correctly
- No rendering issues encountered
- Goose rendering is functioning correctly
- Scene rendering is normal.

Test Outcome

- PASS

Notes

Rendering depends on GPU output and the LibGDX graphics pipeline and therefore must be validated manually.

MT-03 — GameState Transition

Requirement(s)

- FR_END_GAME
- NFR_END_GAME
- UR_ESCAPE_MAZE

Component / System

`GameStateSystem.java; WinScreen.java; GameOverScreen.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)
- Keyboard & Mouse
- Game launched via IDE / Gradle

Preconditions

- Active game session
- The player can lose when time expires

Test Steps

1. Start a new game
2. Allow the player time to expire
3. Observe whether, upon reaching the game's end screen, issues such as being unable to restart or becoming stuck occur.
4. Continue with a new game,
5. Play normally until you complete the game before time runs out.

Expected Result

- Game transitions to Game Over screen
- Player input is disabled
- The buttons on the game's end screen can be interacted with normally.
- Finish the game before time runs out to display the win screen.
- The page displays correctly without garbled characters.

Actual Result

- When time runs out, the game proceeds normally to the end screen.
- The restart button and main menu button are functioning correctly.
- Upon entering the escape door before time runs out, the win screen will appear as normal.
- The page buttons function correctly, and the data text displays properly without garbled characters.

Test Outcome

- PASS

Notes

Game state transitions involve multiple systems and observable effects that cannot be fully validated through unit tests.

MT-04 — Pause Menu Activation and Resume

Requirement(s)

- UR_PAUSE
- FR_PAUSE
- FR_PAUSE_MENU
- NFR_PAUSE

- NFR_PAUSE_MENU_NAVIGATION

Component / System

- `PauseScreen.java`, `GameStateSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)
- Keyboard

Preconditions

- Game is actively running

Test Steps

1. Start gameplay
2. Press the pause key (ESC)
3. Observe pause menu appearance
4. Resume gameplay

Expected Result

- Game freezes immediately
- Pause menu is displayed
- Gameplay resumes correctly

Actual Result

- When the pause button is pressed, the pause screen appears as expected.
- The main menu button and the Resume button on the pause screen are functioning correctly.

Test Outcome

- PASS

Notes

Pause behaviour depends on real-time input handling and game loop control.

MT-05 — Timer Countdown and Expiry

Requirement(s)

- UR_TIME_LIMIT
- FR_TIMER
- NFR_TIMER

Component / System

- `TimerSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game session started
- Timer visible on screen

Test Steps

1. Start a new game
2. Observe the timer countdown

3. Allow the timer to reach zero

Expected Result

- Timer decreases accurately
- Game ends when time expires

Actual Result

- When the timer reaches zero, the game-over screen appears as normal.
- When the timer reaches zero, the player's input normally ceases.

Test Outcome

- PASS

MT-06 — Physics Collision and Boundary Enforcement

Requirement(s)

- UR_BOUNDARIES
- FR_BOUNDARIES

Component / System

- `PhysicsSystem.java`, `PhysicsToTransformSystem.java`

Test Type

- Manual Test

Preconditions

- Player can move within maze

Test Steps

1. Move the player towards walls and obstacles

2. Attempt to pass through boundaries

Expected Result

- Player cannot move through walls
- Movement is constrained correctly

Actual Result

- Players cannot pass through walls when moving towards them.
- When players are at the map boundary, clipping and out-of-bounds issues will not occur.

Test Outcome

- PASS

MT-07 — Player Input Responsiveness

Requirement(s)

- UR_UI
- NFR_PERFORMANCE

Component / System

- `PlayerSystem.java`

Test Type

- Manual Test

Test Steps

1. Use movement controls continuously
2. Observe responsiveness and smoothness

Expected Result

- Player responds immediately to input
- No lag or freezing

Actual Result

- When the player inputs W, S, A, D, the player moves normally with no delay.
- When rapidly inputting multiple defensive player movements, no lag occurs.

Test Outcome

- PASS

MT-08 — Main Menu Navigation and Options

Requirement(s)

- UR_UI

Component / Screen

- `MenuScreen.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)
- Keyboard & Mouse
- Game launched via IDE / Gradle

Preconditions

- Game launches successfully
- Main menu is displayed

Test Steps

1. Launch the game
2. Observe the main menu layou
3. Navigate through menu options using keyboard or mouse
4. Select “Start Game”
5. Observe the main interface switching to the gameplay
6. Observe the game controls guide on the main interface

Expected Result

- All menu options are clearly visible
- Navigation between options works correctly
- Selecting “Start Game” begins the game without errors
- The text guidance section on the main interface is displayed correctly.

Actual Result

- The main interface launches correctly without any misalignment.
- All buttons on the main interface are correctly positioned and aligned.
- Button navigation functions correctly; game launches without issue.
- The text in the game tutorial displays correctly, with no garbled characters or disordered sequence.
- Interface switching functions normally, with no delays or crashes occurring.

Test Outcome

- PASS

Notes

Menu navigation depends on real-time input handling and screen rendering and therefore requires manual testing.

MT-09 — Level Screen Initialisation

Requirement(s)

- UR_ESCAPE_MAZE
- FR_MAZE

Component / Screen

- `LevelScreen.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game launched successfully
- Player selects “Start Game” from menu

Test Steps

1. Start a new game from the main menu
2. Observe the initial level layout
3. Confirm player character spawns correctly
4. Attempt basic movement
5. Observe the generation of in-game items
6. Observe the generation of solid walls within the map

Expected Result

- Level loads without delay or error
- Maze environment is visible
- Player is placed inside the maze and can move
- In-game player entities are generating correctly.
- In-game item generation is functioning correctly.(coffee,goose,check-in code,etc)
- Interior wall generation within the map is functioning correctly.

Actual Result

- Upon entering the game, initialisation proceeded normally without crashing.
- Player entities spawn correctly, with no loading errors or misalignment.
- Following multiple rounds of verification, the generation of props is now fixed and complete, with no missing items.
- The goose's initial spawning state is normal, with no missing or stuck components.
- The map wall generates correctly without any clipping or disappearance issues. The hidden door remains inactive and properly blocks the player.
- Map generation proceeded normally without any missing elements, errors, or inconsistencies.

Test Outcome

- PASS

MT-10 — Camera Follow Behaviour

Requirement(s)

- UR_UI
- UR_BOUNDARIES

Component / System

- `CameraFollowComponent.java`
- `WorldCameraSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game level loaded
- Player can move freely

Test Steps

1. Move the player in different directions
2. Observe camera movement relative to the player
3. Move player close to maze boundaries

Expected Result

- Camera smoothly follows player movement
- The camera will not continue moving once the player reaches the map boundary.
- No jittering or sudden jumps occur

Actual Result

- The camera movement speed is normal, and no camera drift occurs when the player moves.
- The player remains at the centre point of the camera at all times.
- When players reach the map boundary, the camera will stop moving as the player collides with the wall and will not continue moving.

Test Outcome

- PASS

Notes

Camera behaviour is dependent on real-time rendering and player movement, making automated unit testing impractical.

MT-11 — Enemy (Goose) Behaviour

Requirement(s)

- UR_NEGATIVE_EVENTS
- FR_NEGATIVE_EVENTS

Component / System

- `GooseSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game level active
- Goose enemy present in the maze

Test Steps

1. Navigate the player close to the goose
2. Observe goose movement and behaviour
3. Allow the goose to interact with the player

Expected Result

- Goose behaves as a negative event
- Players struck by geese will be knocked back and lose points.
- Game continues running correctly
- After attacking a player, the goose will enter a cooldown period and lose its lock-on.
- Geese within the player's field of vision will pursue the player.

- During pursuit, the goose will disengage and attempt to return to its starting point once the player moves beyond a certain distance.

Actual Result

- The goose's AI behaviour is functioning normally, actively pursuing players when within their field of vision.
- After completing its first attack, the goose enters a cooldown period, losing its lock-on to the player and unable to launch multiple attacks in quick succession.
- When the player moves beyond a certain distance from the goose, it will disengage and attempt to return to its starting point.(There is a probability of causing clipping at corners and model positions, but this is acceptable.)
- When geese attack players, players will move and lose points, with no lag or crashes.
-

Test Outcome

- PASS

MT-12 — Interactable Objects Behaviour

Requirement(s)

- UR_POSITIVE_EVENTS
- FR_POSITIVE_EVENTS

Component / System

- `InteractableSystem.java`
- `InteractableComponent.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game level active
- Interactable objects present

Test Steps

1. Move player towards an interactable object(coffee, pie, bob, etc)
2. Trigger interaction(Pick up the coffee, pick up the pie, pick up the glasses, interact with Bob, pick up the check-in code, interact with the hidden door mechanism.)
3. Observe resulting effect

Expected Result

- Interaction triggers a positive effect
- The effect is applied immediately
- No unexpected behaviour occurs
- Items will vanish upon being picked up.
- Picking up coffee grants a brief speed boost,
- Picking up a check-in code awards points,
- Picking up pies causes them to vanish.
- Picking up Bob will earn you points and trigger a dialogue.
- The concealed door opens normally.

Actual Result

- After picking up the coffee, the acceleration function operates normally without any instances of teleportation or model clipping; the coffee model disappears correctly.
- The pie will disappear normally after being picked up.
- Upon retrieval of the check-in code, the model vanishes, and the player's score increases.
- After picking up Bob's glasses and returning them to him, a dialogue will trigger where Bob expresses his gratitude.
- When the player goes to find Bob without his glasses, a greeting conversation will be triggered.
- When the player triggers the mechanism by passing through the hidden door, the hidden door opens normally. The mechanism's model disappears, with no air walls or model disappearance errors occurring.

Test Outcome

- PASS

MT-13 — Achievement Manager

Requirement(s)

- UR_ACHIEVEMENTS
- FR_ACHIEVEMENTS_SYSTEM
- NFR_ACHIEVEMENTS_VALIDATION

Component / System

- `AchievementManager.java`
- `Achievement.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game level active.
- No achievements have been unlocked.
- The player has just started the maze.

Test Steps

1. Collect all coffees.
2. Collect all slices of pi.
3. Go to the hallway with the geese and let them hit you 10 times.
4. Collect the pink glasses and give them to Bob.
5. Step on the pressure plate.
6. Collect all of the check-in codes.
7. Exit to the main menu and check the achievement screen.
8. Go back into the maze and collect all the coffee.
9. Exit to the main menu, go to the achievement screen and reset the data.
10. Go back into the maze and collect all the coffee.

Expected Result

- Collecting all coffee awards the “Coffee Addict” achievement.
- Collecting all pi slices awards the “Pi Collector” achievement.
- After the 10th goose bite, the player receives the “Bird Food” achievement.
- Once Bob has received the glasses, the player receives the “SYS1 Trauma” achievement.
- Stepping on the pressure plate awards the “CLICK?!?” achievement.
- Collecting all check in codes awards the “Model Student” achievement.
- After completing the previous steps, all achievements should be marked as unlocked.
- You no longer receive the achievement for collecting all coffee as you have already unlocked it.
- The achievements are now locked after resetting data.
- Going back to the maze and collecting the coffee awards the “Coffee Addict” achievement as before.

Actual Result

- Collecting all the coffee gave me the “Coffee Addict” achievement and a blue toast came up to let me know.
- Collecting all pi slices gave me the “Pi Collector” achievement and a blue toast came up to let me know.
- After letting the geese attack for a while, I got the “Bird Food” achievement.
- I didn't get anything for collecting the glasses, but once I gave them to Bob I got the “SYS1 Trauma” achievement.
- I stepped on the pressure plate to open the secret room and was given the “CLICK?!?” achievement.
- Once I collected the last check-in code and my attendance reached 100%,I was given the “Model Student” achievement.
- In the menu, all the achievements were displayed with white text and had their descriptions, showing that they were all unlocked.
- This time, I did not get a toast telling me I unlocked “Coffee Addict” when I got the last coffee.
- Resetting the data caused all the achievement text to go grey, and the descriptions to be hidden. This means they are all locked.

- Collecting the coffee now awards me with the achievement again, with the blue toast to let me know.

Test Outcome

- PASS

MT-14 — Leaderboard

Requirement(s)

- UR_LEADERBOARD
- FR_LEADERBOARD_STORAGE
- FR_LEADERBOARD_DISPLAY
- NFR_LEADERBOARD_PERSISTENCE

Component / System

- `LeaderboardManager.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game level active.
- Have an empty leaderboard.

Test Steps

1. Do a run of the maze and get a really low score.
2. Add this to the leaderboard as “Lower”.
3. Do a run of the maze and get a low score.
4. Add this to the leaderboard as “Low”.
5. Do a run of the maze and get a high score.
6. Add this to the leaderboard as “High ”.
7. Do a run of the maze and get a medium score.
8. Add this to the leaderboard as “Medium”.
9. Do a run of the maze and get a higher score.
10. Add this to the leaderboard as “Higher ”.
11. Go to the leaderboard menu and check the leaderboard.
12. Do another run of the maze and get the lowest score yet.

13. Check if it lets you add this to the leaderboard.
14. Do a run of the maze and get the highest score.
15. Add this to the leaderboard as “Highest ”.
16. Check if “Lower” is still on the leaderboard

Expected Result

- The really low score will be added to the top of the empty leaderboard.
- The low score will be added to the top of the leaderboard, as it is higher than lower. This causes “Lower” to go down a rank.
- The high score will be added to the top of the leaderboard, as it is higher than low. This causes the other entries to move down a rank.
- The medium score will go in between high and low. High will stay where it is, but the the other entries move down a rank.
- The higher score will be added to the top of the leaderboard. The other entries move down a rank.
- The leaderboard should be full. Containing the five entries: Higher, High, Medium, Low, Lower in that order.
- The game will not let you add the lowest score to the leaderboard as it is not in the top 5.
- The new highest score will be added to the top of the leaderboard, making all the other entries to move down a rank. This will remove “Lower” from the leaderboard.

Actual Result

- The first really low score “Lower” of 186 was placed at rank 1 as there are no other scores.
- The next score “Low” was 293 and placed at rank 1, moving “Lowest” down to rank 2
- The next score “High” was 400. This is the new best, so it was placed at rank 1. This moves “Low” to rank 2, and “Lower” to rank 3.
- The next score “Medium” was 376. This is not as high as 400, so “High remained in rank 1. However, “Medium” became rank 2, pushing “Low” to rank 3, and “Lower” to rank 4.
- The next score “Higher” was 581. This is the new best, so it was placed at rank 1. This moves “High” to rank 2, “Medium” to rank 3, “Low” to rank 4, and “Lower” to rank 5.
- The leaderboard is full, with the scores 581, 400, 376, 293, 186 in the correct order.

- The next score was 162. This is lower than the score in rank 5 (186), so the game does not give me the option to add this new score to the leaderboard.
- The next score “Highest” was 661. This is the new best, so it was placed at rank 1. This moves “ “Higher” to rank 2, “Medium” to rank 4, “Low” to rank 5, and “Lower” is removed from the leaderboard.

Test Outcome

- PASS

MT-15 — Roboto Font Generator

Requirement(s)

- UR_UI

Component / System

- `FontGenerator.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Menu screen active.
- Screen contains at least one button, with all buttons having

  Font(baseSize: 24, colour: White).

Test Steps

1. Load the menu screen and check the text on the buttons.
2. Close the game and edit the code so that the button font has base size 48
3. Check if the text size has changed.
4. Close the game and edit the code so that the button font has base size 12
5. Check if the text size has changed.
6. Close the game and edit the code so that the button font is back to 24, but change the colour to something else.
7. Check if the text is the correct colour.

Expected Result

- The text size will scale with the value, roughly halving and doubling when it should.
- The colour will change depending on which colour is set.
- All buttons will update, always looking the same.

Actual Result

- Upon loading the menu all buttons loaded correctly, displaying white text inside rectangular grey boxes.
- When the size was changed to 48, the text roughly doubled in size, such that the text was now bigger than the grey boxes. The text remained the same colour.
- When the size was changed to 12, the text roughly halved in size from what it was at the start. The text remained the same colour.
- When the colour was changed from white to red, all of the buttons text updated to the same red.

Test Outcome

- PASS

MT-16 — Raspberry Pi Collect

Requirement(s)

- UR_HIDDEN_EVENTS

Component / System

- `InteractableComponent.java`
- `PiSystem.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game level active.
- 5 slices of Pi are loaded onto the map..
- The player has not collected any Pi.

Test Steps

1. Collect a slice of pi
2. Check if the UI in the bottom left updates to show you have collected it, and that the slice has been removed.
3. Repeat 1 and 2 until all five slices are collected.
4. Finish the maze and see how that affected your score..
5. Restart the maze and collect one slice.
6. Finish the maze and see how that affected your score.
7. Restart the maze and collect one more slice.
8. Finish the maze and see how that affected your score
9. Repeat 7, 8 until you finish a run with 4 slices.

Expected Result

- The circle in the bottom left corner will fill up with slices until the final slice is collected and the circle is complete.
- The slices will not affect your score, or appear on the summary, unless all 5 were collected. If all were found, +100 is added to the score.

Actual Result

- Upon collecting the first slice, the ui updated to show one segment filled in.
- After each slice was collected, the segments filled in.
- When the final slice was collected, the circle was completely filled in.
- When I exited the maze, I gained +100 score for completing the pi.
- Finishing the maze with any other number of slices did not affect the score, and didn't show up in the score summary at the end.

Test Outcome

- PASS

MT-17 — MazeGame Launcher

Requirement(s)

- UR_UI

Component / System

- `MazeGame.java`

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game is off

Test Steps

1. Launch the file MazeGame.java
2. Press to access achievements
3. Exit from achievements
4. Press to access leaderboard
5. Exit from leaderboard
6. Press to start the game
7. Pause the game and exit
8. Press Quit to exit the game

Expected Result

- The game launches.
- I am able to access achievements, leaderboard, play the game and exit the game

Actual Result

- Upon running the file, pop up window shows, displaying the starting menu screen with the ability to play the game, access leaderboard and achievements and Quit the game
- Pressing leaderboard shows the leaderboard
- Pressing achievements shows the achievements
- Pressing start game, starts the game
- Pressing Quit, closes the program

Test Outcome

- PASS

MT-18 — Exiting the Maze

Requirement(s)

- FR_WAY_OUT
- UR_ESCAPE_MAZE

Component / System

- N/A

Test Type

- Manual Test

Test Environment

- Desktop PC (Windows)

Preconditions

- Game is running

Test Steps

1. Navigate through the maze
2. Find the exit
3. Walk up to the door
4. Game should finish, saying player escaped the university

Expected Result

- The game launches.
- Player finds exit
- Walks to exit
- Game announces the player escaped and the player score

Actual Result

- The game launches.
- Player finds exit
- Walks to exit
- Game announces the player escaped and the player score

Test Outcome

- PASS