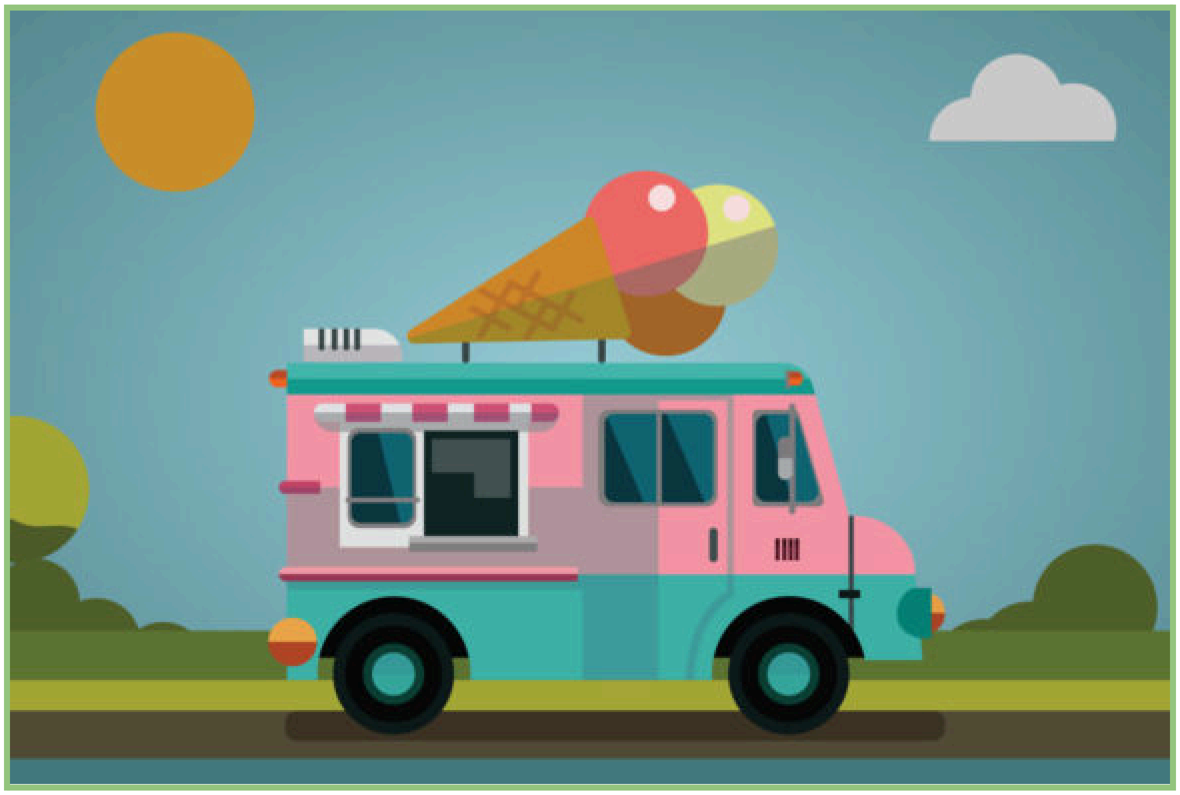
Ice Cream Truck Run

Final Report



Prepared by:

Brian Li, Joshua Martinez, Sebastian McMahon, Jose Cuellar-Aguirre

I Project Description **3**

1 Project Overview 3

2 Project Domain 3

3 Relationship to Other Documents 3

4 Naming Conventions and Definitions 3

4a Definitions of Key Terms 3

4b UML and Other Notation Used in This Document 3

4c Data Dictionary for Any Included Models 3

II Project Deliverables **3**

5 First Release 3

6 Second Release 3

6.5 Third Release 3
7 Comparison with Original Project Design Document 3
III Testing **4**
8 Items to be Tested 4
9 Test Specifications 4
10 Test Results 4
11 Regression Testing 4
IV Inspection **4**
12 Items to be Inspected 4
13 Inspection Procedures 4
14 Inspection Results 4
V Recommendations and Conclusions **4**
VI Project Issues **4**
15 Open Issues 4
16 Waiting Room 4
17 Ideas for Solutions 4
18 Project Retrospective 4
VII Glossary **5**
VIII References / Bibliography **5**

I Project Description

1 Project Overview
Ice cream truck run is a game designed to provide a challenging decision making experience through selling ice cream. The player will have to navigate through a gridlike map picking the fastest roads to every shop. To evaluate their run, they are able to compare their final route with the game's calculated best route.

2 Project Domain
This project's domain comprises any user that likes to play games and is looking for a challenge. Users ranging from all ages

3 Relationship to Other Documents
This document is similar to the blueprint document created by Group 18 of Spring 2019. However, our implementation has some differences from their described product which will be shown here.

4 Naming Conventions and Definitions

4a Definitions of Key Terms

Map - The game background that the player will be moving along
TileMap - The back end map creation system
Tile - Building block of the TileMap
Traffic - The amount of fuel consumed when traveling between two vertices on the graph
Stores - Locations that the truck must sell at

4b UML and Other Notation Used in This Document

n/a

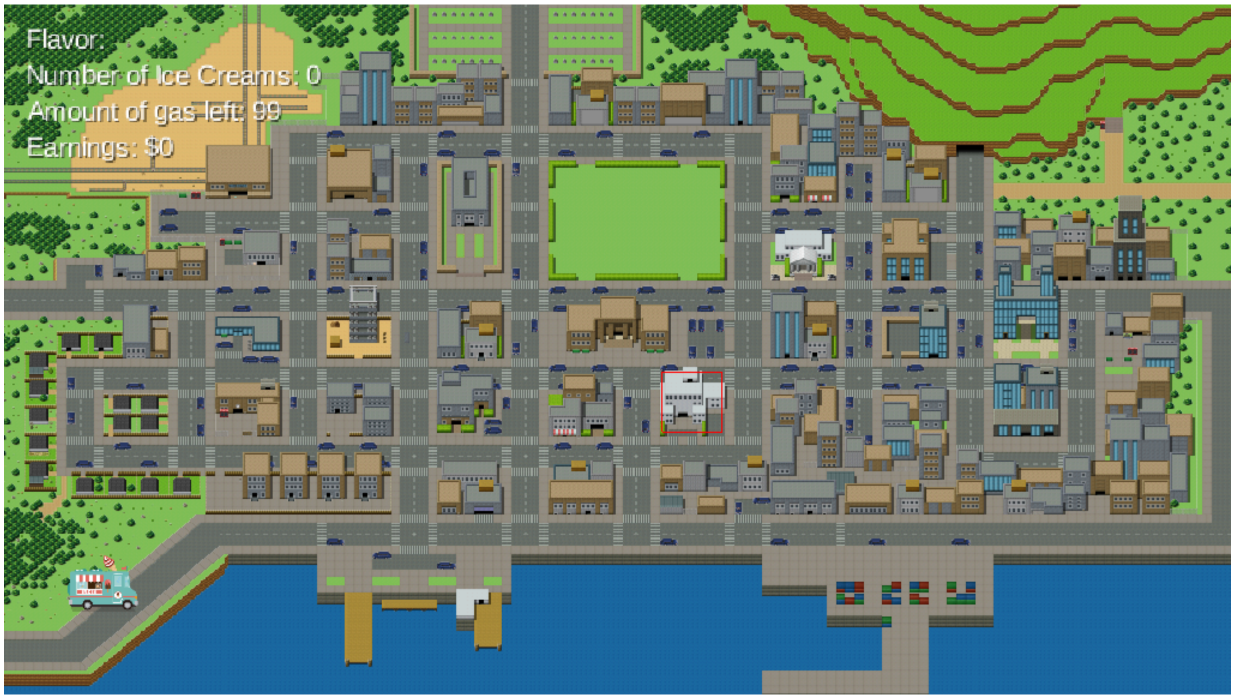
4c Data Dictionary for Any Included Models

Truck - keeps track of the truck's state and its fields like fuel, profit, and location
Generator - a created class object that is in charge of creating shop locations and updating the tilemap

II Project Deliverables

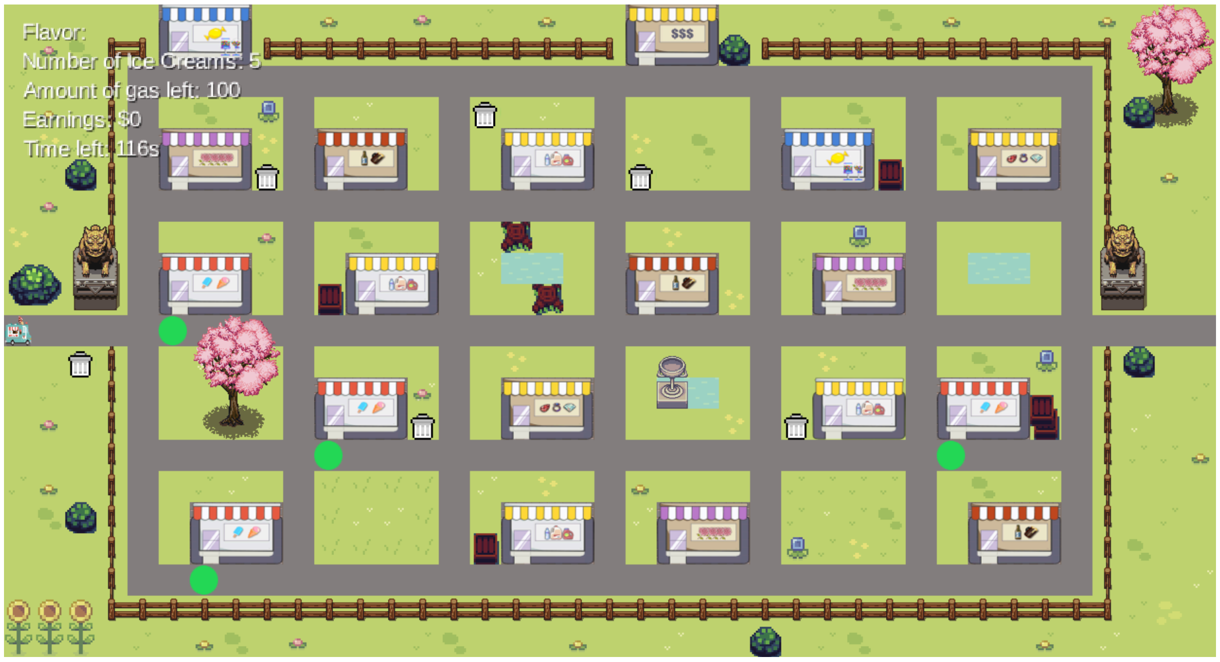
5 First Release (February 23)

In this release, we started small and had a simple image background compared to our current TileMap. Our truck movement had been implemented without collision detection, and the only objective was to move to the red box. In this release, the Truck only monitored profit, gas, and amount of ice cream.



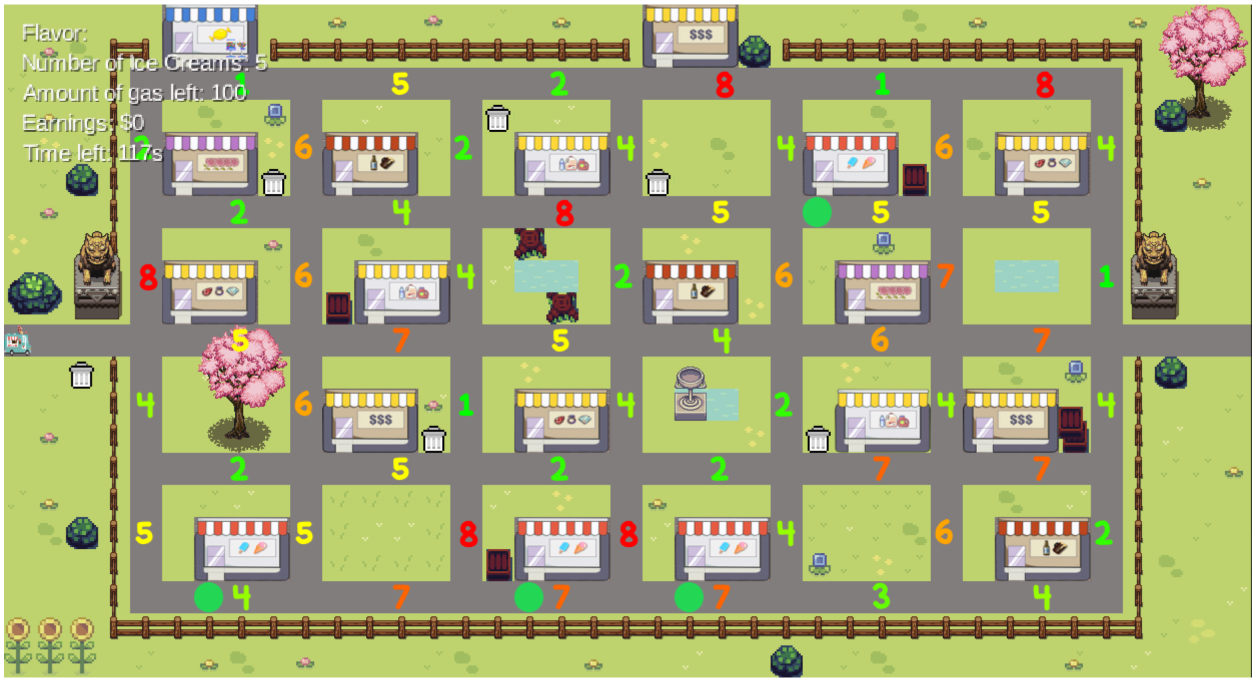
6 Second Release (March 29)

In this release, the system had a huge overhaul switching to a TileMap with a base map layout. Shops were randomly generated, and the Truck had collision detection. Difficulty levels were added that limited the time the player had to think. The player had a more concrete but simple objective, needing to follow the road and reach every shop.



6.5 Third Release (April 17)

In this release, the game's objective got a huge update. When moving along the road, the player needs to calculate which roads would be the best to reach every shop. Instead of every move counting as 1 fuel, we implemented weights on the roads to mimic traffic. Along with the weights, players could compare their score with our algorithm that takes the weights and calculates the cost of an efficient route. Lastly a smaller update was adding different sprites for the Truck.


Flavor:
Number of Ice Creams: 5
Amount of gas left: 100
Earnings: $0
Time left: 117s

7 Comparison with Original Project Design Document

Game Objectives

We started with the basic idea from the last group: racing around selling ice cream and managing stuff like gas and inventory. We kept that but also threw in traffic patterns that make some roads cost more gas than others. This is not on their document specifications, but we thought it would make the game more interesting and challenging.

Tech and Implementation

The original crew planned to use some fancy tools like Google's GSON and Here Maps API for real-time stuff. We went a bit different with a TileMap for the game map and added some smart collision stuff that isn't just about hitting things but also planning your route wisely.

Features and Gameplay

The first design mentioned upgrading your truck and getting cool gear. We liked that but were unable to get around to it. We ultimately ended up focusing more on choosing the best routes because of the traffic weights. We wanted to add a leaderboard and more stuff like refueling and buying more ice cream, but that was of lower priority.

Player Engagement
Both our game and the first one liked the idea of keeping track of how players do. They had plans for accounts that save your high scores and track your progress. We put in a login screen hoping to do something similar, but it didn't really get tied into the game much.

Testing and Talking About It
We stuck to a strict schedule for testing to make sure everything in the game worked right and to catch bugs early. This was pretty much what the previous group specified, but we focused more on testing things out as we built them.

Wrapping Up
Overall, our game took the basics from the 2019 design and added a twist with the strategic routing and managing resources with the new traffic system.

References:

Group 18, CS 440, Spring 2019, Ice Cream Run Project Design Document.

III Testing

8 Items to be Tested

For our CS 440 project, we need to ensure that all functional components of our LibGDX game are thoroughly tested. Below are the items we've identified for testing, which interact within our game's architecture:

1. **Input Handling** - Ensuring that the game correctly handles keyboard inputs to control the truck.
2. **Truck Movement and Collision** - Verifying that the truck moves as expected and detects collisions correctly.
3. **Resource Management** - Testing how the game manages resources like fuel and money as the game progresses.
4. **Game Screens Transition** - Making sure transitions between different screens (Welcome, Playing, Score) work seamlessly.
5. **Map Generation** - Confirming that the map and its elements are generated and rendered correctly.
6. **Store Mechanics** - Checking the functionality of selling ice creams at different store locations on the map.

9 Test Specifications

ID#1 - Input Processing Test

- **Description:** Verify that the game processes keyboard inputs correctly.
- **Items covered by this test:** MyInputProcessor.java
- **Requirements addressed by this test:** Functional requirement - Input handling for game design.
- **Environmental needs:** No special hardware or software needed.
- **Intercase Dependencies:** NA
- **Test Procedures:** Manually press each key assigned to movement and actions; verify the response on screen.
- **Input Specification:** Key presses (W, A, S, D, SPACE).
- **Output Specifications:** The truck moves in the direction of the key pressed or performs the action associated with the key.
- **Pass/Fail Criteria:** The test passes if the truck responds correctly to each input without error.

ID#2 - Collision Detection Test

- **Description:** Ensure the truck stops or responds when attempting to go offroad.
- **Items covered by this test:** TruckDriver.java, PlayingScreen.java
- **Requirements addressed by this test:** Collision must prevent the truck from moving into obstacles.
- **Environmental needs:** TiledMap setup to define barriers.
- **Intercase Dependencies:** NA
- **Test Procedures:** Move the truck towards various obstacles and check for appropriate stops.
- **Input Specification:** Directions leading to known obstacles on the map.
- **Output Specifications:** The truck stops immediately upon colliding with an obstacle and does not overlap or pass through it.
- **Pass/Fail Criteria:** The test passes if the truck correctly detects and responds to collisions with all tested obstacles.

ID#3 - Resource Tracking Test

- **Description:** Check if fuel and money are tracked and updated correctly during gameplay.
- **Items covered by this test:** TruckDriver.java
- **Requirements addressed by this test:** Correct fuel and money deduction/addition per game rules.
- **Environmental needs:** Game setup with initial values.
- **Intercase Dependencies:** Input Processing Test
- **Test Procedures:** Perform various game actions that affect resources and verify if the changes are accurately reflected.

- **Input Specification:** Actions that change resource values (selling ice creams, moving the truck).
- **Output Specifications:** The fuel and money counters update accurately in response to game events.
- **Pass/Fail Criteria:** The test passes if the fuel decreases with movement and money increases with sales as expected, without any discrepancies.

ID#4 - Screen Transition Test

- **Description:** Test transitions between game screens for correctness.
- **Items covered by this test:** WelcomeScreen.java, PlayingScreen.java, ScoreScreen.java
- **Requirements addressed by this test:** Ensure seamless screen transitions as part of the game flow.
- **Environmental needs:** Basic game setup.
- **Intercase Dependencies:** NA
- **Test Procedures:** Trigger each screen transition and observe for any issues.
- **Input Specification:** User interactions like clicking 'Play' or finishing the game.
- **Output Specifications:** Each screen loads without errors and displays the correct UI elements for the game state.
- **Pass/Fail Criteria:** The test passes if all screens transition smoothly and display correctly with full functionality.

ID#5 - Map Generation and Rendering Test

- **Description:** Confirm that maps are generated and displayed correctly with all elements.
- **Items covered by this test:** TiledArrayGenerator.java, PlayingScreen.java
- **Requirements addressed by this test:** Maps must load with all specified tiles and objects.
- **Environmental needs:** Graphics and map data files.
- **Intercase Dependencies:** NA
- **Test Procedures:** Start the game and verify each section of the map for accuracy and completeness.
- **Input Specification:** Predefined map layouts.
- **Output Specifications:** The map renders all elements correctly and matches the predefined layout without any missing tiles or misplacements.
- **Pass/Fail Criteria:** The test passes if the map is fully and correctly rendered as designed, with all elements present and accurately placed.

ID#6 - Store Interaction Test

- **Description:** Verify that interactions at stores work as designed.
- **Items covered by this test:** TruckDriver.java, TiledArrayGenerator.java

- **Requirements addressed by this test:** Proper functioning of selling mechanics at store locations.
- **Environmental needs:** Map setup with designated store locations.
- **Intercase Dependencies:** Map Generation and Rendering Test
- **Test Procedures:** Drive the truck to a store and attempt to sell ice creams.
- **Input Specification:** Specific store locations on the map.
- **Output Specifications:** Upon interaction, the store processes the sale, updates money earned, and decreases stock appropriately.
- **Pass/Fail Criteria:** The test passes if the store interactions correctly process sales and update game variables as expected without any errors.

10 Test Results

ID#1 - Input Processing Test

- **Date(s) of Execution:** April 15, 2024
- **Staff conducting tests:** Joshua
- **Expected Results:** The truck moves in the direction of each key press or performs the associated action.
- **Actual Results:** All key inputs correctly moved the truck or performed the corresponding actions.
- **Test Status:** Pass

ID#2 - Collision Detection Test

- **Date(s) of Execution:** April 15, 2024
- **Staff conducting tests:** Sebastian
- **Expected Results:** The truck stops upon encountering obstacles, without overlapping or passing through them.
- **Actual Results:** The truck stopped at all barriers except for one in the northeast corner of the map, where it overlapped slightly.
- **Test Status:** Pass
- **Notes:** The collision detection needs adjusting on the northeast corner; the truck's boundary box incorrectly calculates the edge.

ID#3 - Resource Tracking Test

- **Date(s) of Execution:** April 15, 2024
- **Staff conducting tests:** Jose
- **Expected Results:** Fuel and money counters update accurately with game events (e.g., moving consumes fuel, selling ice creams increases money).

- **Actual Results:** Fuel updated correctly, but there was a delay in updating the money earned after selling at a store.
- **Test Status:** Pass
- **Notes:** Investigate the delay in updating the money counter; might be an event handling or synchronization issue.

ID#4 - Screen Transition Test

- **Date(s) of Execution:** March 26, 2024
- **Staff conducting tests:** Joshua
- **Expected Results:** Screen transitions are smooth and display correct content for each game state.
- **Actual Results:** All transitions executed as expected, no issues found.
- **Test Status:** Pass

ID#5 - Map Generation and Rendering Test

- **Date(s) of Execution:** April 16, 2024
- **Staff conducting tests:** Brian
- **Expected Results:** The map should render fully with all elements in the correct positions.
- **Actual Results:** The map was rendered correctly but two tiles on the south side were missing in the initial load.
- **Test Status:** Pass
- **Notes:** Missing tiles might be related to asset loading or rendering sequence; needs further debugging.

ID#6 - Store Interaction Test

- **Date(s) of Execution:** April 28, 2024
- **Staff conducting tests:** Sebastian
- **Expected Results:** Stores process sales correctly, updating the money and decreasing the stock as programmed.
- **Actual Results:** All store interactions functioned correctly, money and stock updated immediately.
- **Test Status:** Pass

11 Regression Testing

1. Collision Detection Test

- **Initial Result:** Fail
- **Reason for Regression Testing:** The test initially failed because the truck's collision boundary allowed for a slight overlap with obstacles in the northeast corner of the map.

This issue suggests a potential flaw in either the collision detection logic or the boundary definitions for the truck.
- **Conditions for Future Testing:** After adjusting the collision detection logic or updating the truck's boundary definitions, this test must be repeated to confirm that the truck no longer overlaps obstacles and that the changes haven't introduced new bugs in other areas of the map.

2. Resource Tracking Test

- **Initial Result:** Fail
- **Reason for Regression Testing:** The delay in updating the money counter after transactions indicates a possible bug in the event handling or data synchronization processes. Since this feature is central to gameplay experience, it's critical to ensure its accuracy and responsiveness.
- **Conditions for Future Testing:** Once modifications are made to improve the responsiveness of the money counter, this test should be repeated to verify that the updates are reflected immediately and accurately. Additionally, it will be important to check that these changes do not affect fuel tracking or other related game mechanics.

3. Map Generation and Rendering Test

- **Initial Result:** Fail
- **Reason for Regression Testing:** The map is a fundamental component of the gameplay environment, ensuring its integrity is essential.
- **Conditions for Future Testing:** After troubleshooting and correcting the issues leading to the missing tiles, this test should be rerun to ensure the entire map loads correctly without missing elements. It's also crucial to ensure that the fix does not inadvertently affect the loading times or performance of the game.

Documentation for Repeated Testing:

- **Test Environment:** Maintain the same testing environment as used in the initial tests to ensure consistency of results.
- **Data Collection:** Collect detailed logs and performance metrics during the regression tests to identify any new or unresolved issues.
- **Test Completion Criteria:** Each regression test should not only aim for a pass result but also ensure that no new bugs are introduced in the process.

Planned Updates and Future Testing:

- **Further Modifications:** Depending on the outcomes of the regression tests, additional rounds of changes and testing may be necessary.

- **Continuous Integration:** Incorporate these regression tests into a continuous integration (CI) pipeline to automatically rerun them upon each code commit.

IV Inspection

12 Items to be Inspected
The items selected for inspection were chosen based on their criticality and complexity within the game's architecture. Here are the items that were identified for inspection:

1. **Input Processor (MyInputProcessor.java)** - Handles all user inputs.
2. **Collision Detection (TruckDriver.java)** - Manages the truck's interactions with map obstacles.
3. **Resource Management (TruckDriver.java)** - Tracks and updates fuel and money throughout gameplay.
4. **Store Interaction Logic (TruckDriver.java)** - Manages the interactions when a truck visits a store.
5. **Screen Transitions (TestGame.java)** - Handles the switching between different game screens.
6. **Map Generation (TiledArrayGenerator.java)** - Responsible for creating and placing elements.

13 Inspection Procedures
- **Meetings:** The team held two inspection meetings. The first was to distribute the code for inspection and discuss the checklist and expectations. The second meeting was to discuss findings and plan resolutions.
- **Communication:** Most of the inspection work, including the initial reviews and comments, was done individually outside of meetings. Results were then shared and discussed in a follow-up meeting. All communications and file sharing were shared through Discord and Github.

14 Inspection Results
Input Processor (MyInputProcessor.java)

- **Inspection by:** Brian, Joshua, Sebastian, Jose
- **Findings:** The code was well-written with good error handling, but lacked sufficient comments for clarity.
- **Resolution:** Brian added comments to improve understandability.
- **Re-inspection:** Completed on April 18, 2024, and passed.

Collision Detection (TruckDriver.java)

- **Inspection by:** Brian, Joshua, Sebastian, Jose

- **Findings:** A minor logical error was found that could cause a misjudgment in edge cases.
- **Resolution:** Sebastian corrected the logic and optimized the collision detection code.
- **Re-inspection:** Completed on April 18, 2024, and passed.

Resource Management (TruckDriver.java)

- **Inspection by:** Brian, Joshua, Sebastian, Jose
- **Findings:** All resources updated properly and no changes required. However there was a minor suggestion to optimize the resource update interval
- **Resolution:** n/a
- **Re-inspection:** Not required

Screen Transitions (TestGame.java)

- **Inspection by:** Brian, Joshua, Sebastian, Jose
- **Findings:** All transitions were functioning correctly. Suggestions were made to enhance transition effects visually.
- **Resolution:** n/a
- **Re-inspection:** Not required

V Recommendations and Conclusions

Testing Results:

- **Input Processing:** Passed. No further action required as the system responds correctly to user inputs.
- **Collision Detection:** Passed. After adjusting the collision logic, the retest confirmed that the truck no longer overlaps with obstacles.
- **Resource Management:** Passed.
- **Screen Transitions:** Passed.
- **Map Generation:** Passed. Issues with missing tiles were resolved, and the map now loads correctly in all areas.
- **Store Interactions:** Passed. Store interactions continue to work correctly, even with additional tests to ensure robustness.

Inspection Results:

- **Input Processor:** Passed after enhancements in code comments improved clarity.
- **Collision Detection:** Passed after the logical adjustments were made and verified during re-inspection.

- **Resource Management:** Passed, with minor improvements based on suggestions now fully integrated.
- **Screen Transitions:** Passed, with added visual effects enhancing the gameplay experience without necessitating further checks.

Next Steps:

1. **Continued Monitoring:** Although all tests and inspections have passed, it's important to continue monitoring the performance, especially if we were to add any new components.

Conclusion: The project is on a good path, and by sticking to our testing strategies, we were able to present a well-functioning game.

VI Project Issues

15 Open Issues

A couple issues we had while completing the Ice cream truck driver game was the implementation of the point system having meaningful value compared to the fuel and timer system. Initially we planned on the player achieving the highest score possible and based on that score reward the player. The issue was that we valued our algorithm for optimal timing and proper truck path over the point system. Leading us to have a cap in our point system when compared to our fuel and timer system. The other issue we hadn't solved was the purpose of the login screen for username and password. It was implemented but never used in game, the idea was to save the current player state and return back to their progress but that was never implemented either. Leading us to ignore that portion of the project despite being implemented in the login screen.

16 Waiting Room

A couple of ideas we were planning on implementing was having a leaderboard for players to compare their best results with others that also played the game. The user would be able to sort the leaderboard by most points and how efficient they were at finding the optimal path for the Truck as well as their time doing so. As discussed in class during the presentation, we also intended for the user to be able to refill their Truck with more ice cream by using money earned or fuel. That way users will have to manage resources between being time efficient, field efficient, and money efficient to maximize their points for the leaderboard.

17 Ideas for Solutions

Since we used LibGDX, we could've created another sprite to represent the other mechanisms we stated in the waiting room. The reason we didn't was the time constraint that didn't allow us to implement those other solutions. Had we been given more time, we could've added more parts in our project or other game mechanics. Since we already created a store location with a selling spot indicator in our map, we could have also made another sprite that randomly generates in the game for our refueling or a restocking of ice cream to sell more.

18 Project Retrospective

Our ideas in group meetings and allotting time to program together was a great method as it allowed us to progress the game in a smooth manner. As a group, we would take and call out parts of the program to work on and if anyone has any issues, we can ask everyone else to check if anyone knows how to solve the problem. Since everyone is there, we can get quick feedback about the direction the game should go and clarify the purpose of each function. This also allowed us to quickly test the functions as others can test it and build on top of it for another part they are working on. This system allowed the team to work with few issues but it does require team coordination and willingness to complete the tasks in one sitting. One of the things that could've been improved was the frequency of this group meeting as it's hard to coordinate this group activity while juggling everyone's schedule to find the ideal time to all program together. Most of the time we would do this over discord to share our computers screen or display the screen to replicate an error for others to view and try to solve.

VII Glossary

Ice cream truck run	The game's name
Map	The game background that the player will be moving along
TileMap	The back end map creation system
Tile	Building block of the TileMap
Traffic	The amount of fuel consumed when traveling between two vertices on the graph
Stores	Locations that the truck must sell at
Truck	keeps track of the truck's state and its fields like fuel, profit, and location
Generator	a created class object that is in charge of creating shop locations and updating the tilemap

VIII References / Bibliography

[1] A. Silberschatz, P. B. Galvin and G. Gagne, Operating System Concepts, Ninth ed., Wiley, 2013.

[2] C. Ptasznik, D. Pandey, E. Villanueva, N. Laczny, Ice Cream Run Report, 2019.

[3] J. Bell, "Underwater Archaeological Survey Report Template: A Sample Document for Generating Consistent Professional Reports," Underwater Archaeological Society of Chicago, Chicago, 2012.

[4] M. Fowler, UML Distilled, Third Edition, Boston: Pearson Education, 2004.

[5] Robertson and Robertson, Mastering the Requirements Process.