Traffic Jam: Scaffolded Game Development to Teach Object-Oriented Programming

Osvaldo Jiménez
ojimenez@pacific.edu
University of the Pacific
Stockton, United States

Sebastian Dziallas
sdziallas@pacific.edu
University of the Pacific
Stockton, United States

Course Special Issue
Programming Language Java
Knowledge Unit Software Development Methods
CS Topics Object-Oriented Programming, Collections / Data Structures, Software Development and Design
Resource Type Assignment

Synopsis

Traffic Jam is an assignment designed for a CS3-style course that is meant to (re-)introduce students to larger programming projects. Students implement a grid-based puzzle game based on the sliding block puzzle game Rush Hour. In the game, a player must slide adjacent cars facing horizontally or vertically in order to dislodge a particular car and move it to the other side of the grid.

This assignment consists of scaffolded instructions, UML diagrams, and starter files that guide students through the process of creating the game in four parts. This allows students to build on their work and to work on increasingly complex goals, from a text-based version to a graphical user interface in Java using the ACM Java library.

For this assignment, students should be familiar with arrays, methods, references, and classes. The project aims to help students gain more experience and proficiency with classes and object-oriented design in Java, as well as with event handlers and interactivity. Students learn how to read existing code and supplement it with new features to build a prototype of the game.

Upon completion of this assignment, students will be able to:

- Design and implement robust Java classes to model complex game entities.
- Manipulate 2D-arrays and manage object references to represent and update a grid-based game state.
- Apply the Model-View-Controller (MVC) pattern to separate the underlying game logic from both console-based and graphical user interfaces.
- Construct event-driven graphical interfaces using external libraries to handle user inputs (e.g. mouse dragging and clicking).

ACM EngageCSEdu, December 2025.

ACM ISBN 979-8-4007-2419-0/2025/12.
https://doi.org/10.1145/3786356

Keywords

Game Development, CS3, Object-Oriented Design, State Management, Event-Driven Programming

ACM Reference Format:

Osvaldo Jiménez and Sebastian Dziallas. December 2025. Traffic Jam: Scaffolded Game Development to Teach Object-Oriented Programming. In ACM *EngageCSEdu*. ACM, New York, NY, USA, 3 pages. https://doi.org/10.1145/3786356

1 Engagement Highlights

This project aligns with the NCWIT engagement framework by using meaningful and relevant content, giving effective encouragement, and offering student-centered assessments.

In terms of meaningful and relevant content, this assignment gives students the opportunity to create a prototype of a game that mirrors what they might find in an app store and play on their phone. In fact, the idea for this assignment grew out of a student project from an earlier iteration of the course.

Regarding giving effective encouragement, the project aims to make it okay to make mistakes. As the four parts of the assignment build on each other, it can be challenging for students when issues from earlier parts affect later parts of the project. We address this by allowing students to modify code that was completed in earlier parts. Additionally, in the final part of the project, students have the option to use their own code or to build on an existing functional solution of the earlier parts.

Finally, in terms of offering student-centered assessments, we aim to encourage students to seek help. Traffic Jam is designed to help students identify areas where they might need further support and students are regularly encouraged (both in the assignment and in class) to seek additional support from TAs, through office hours, or in tutoring. We also provide additional asynchronous resources for students who may be intimidated by office hours, such as narrated PowerPoint slides, videos (e.g. on null pointer exceptions), a document with frequently asked questions, and labs to introduce interactivity in the graphical interface (included here as well).

2 Recommendations

The Traffic Jam assignment typically takes (at least) four weeks, which translates to one week for each of its four parts, including time to work on it both inside and outside of class.

Students at our institution study CS1 and CS2 in a different programming language (C++), so for many of them this is the

first time that they encounter Java. We also have a sizeable population of transfer students and, for them, this is often one of the first courses they take at our institution. This means that the students in this course tend to have a wide range of programming experiences.

Most of the students in the course have little to no familiarity with Java in particular, with most having only seen C++ or Python. The decision to use a different programming language in this course was a deliberate choice in order to help prepare students for future experiences with new programming languages, e.g. in the workplace. The most common struggles students encounter center around object-oriented programming, so we would recommend providing additional or optional materials that students can seek out to support their switch to Java (if applicable).

Finally, this course is designed to provide students with a project-based experience. However, our experience has been that having students choose their projects at the beginning of the semester can be challenging. Therefore, we created this individual assignment to give students an example of what a project might look like before they begin working on their own team projects later in the course. Similar to a summer book given to students before they come to campus, this assignment is then meant to build a shared understanding and language between the students in the course.

3 Description of Assignment Stages

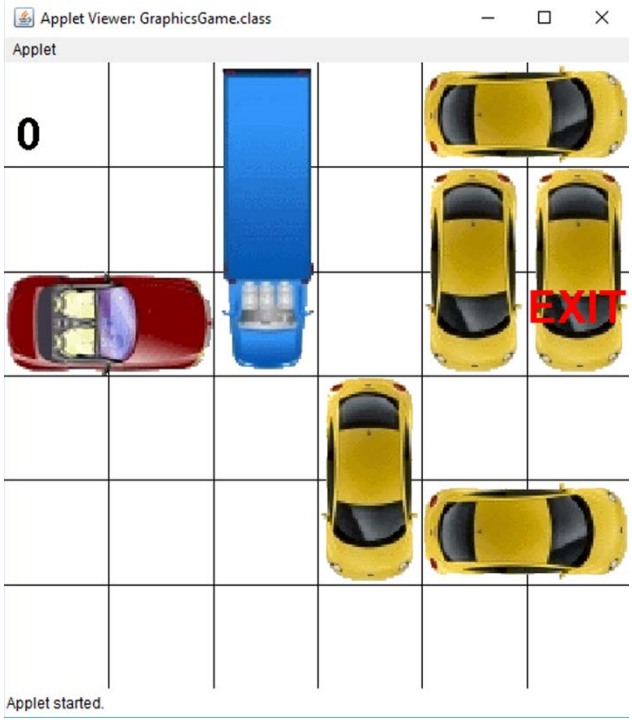

Figure 1: Screenshot of the Traffic Jam Game

3.1 Location & Vehicle

In the first of the four stages of this assignment, the focus is on helping students to get acquainted with basic Java classes and the assignment. Students are asked to work first on a small *Location* class which represents a space on the board in Figure 1 shown above. This class resembles a *Point* class in terms of functionality and requires the implementation of getters and setters. Afterwards, students pair this class with a slightly larger *Vehicle* class, which represents a vehicle on the board. This class must hold a *Location* among a few more variables and additional methods. Several of these methods require more elaborate logic which is used in subsequent stages. We also provide students with initial tests so that they can verify that their two classes are working.

3.2 Board

In the second stage, we guide students to connect classes more with arrays by introducing a new *Board* class which they use to store pointers to vehicles on a 2D board. We chose to have students write code with basic arrays instead of using Java's collections to leverage students' familiarity with arrays from their prior CS1/2 courses. One peculiar design decision here was to have all the spaces on a board that a vehicle occupies point to the same Vehicle object. With this decision, students work on a method called *locationsOn* which, given a vehicle, returns an array of Locations that the *Vehicle* would occupy on that *Board*. Having such a function along with using pointers makes moving vehicles follow OOP principles and introduces students to an alternative OOP design for the game. This emphasizes relying on object references instead of working with index arithmetic. Students once again receive test code to give them initial feedback as to how their board is working, however that test code is not exhaustive.

3.3 Console

We connect this basic infrastructure with a console version of the game, which students implement in the third stage of the assignment. The main theme in this phase is to leverage the existing classes to build a complete game. It focuses on a *Level* class, that serves mostly as an intermediary to the *Board*, and a *ConsoleGame* class that mimics the input and output style of traditional console-based programs. Other than the idea that *Level* serves as that connection between *ConsoleGame* and *Board*, students often report the third stage not being as difficult as the second stage. Having this break tends to serve as an opportunity for students to catch up and to debug their earlier implementations should they wish to do so. Part of our goal during this phase is for students to assess where they are, both in terms of the assignment and their Java skills more broadly, with the hope that they will continue developing in ways they find most beneficial.

3.4 Graphics

Finally, in the fourth stage, students work on a graphical wrapper using the ACM Java library. This stage is designed to help students learn about events and to begin appreciating the idea of a Model-View-Controller Design (see Figure 2 below), since their *Board* and *Vehicle* model does not change in the last stages. For many students, this may be first time they work with events and graphical interfaces. In this phase, students implement mouse drag handlers and call existing game functions to better understand how different classes and existing logic can integrate. With this stage happening 4-5 weeks into the course, we also provide additional in-class labs to guide students through topics such as mouse events and event handlers in the ACM Java library. This last stage is meant to assess whether or not students can use the ACM Java library to create a visual version of the Traffic Jam game.

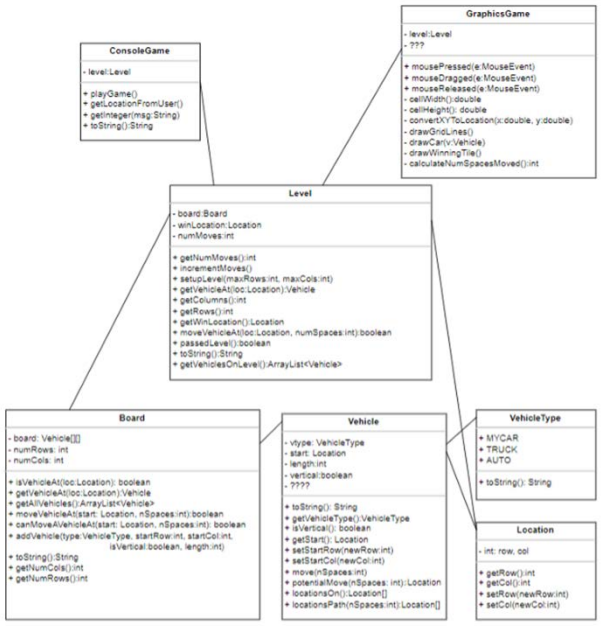

Figure 2: UML Diagram of the Traffic Jam Game

4 Assessment

Each of the four parts of the assignment is graded on a rubric that assesses the implementation and functionality of the code. For the second stage, this also includes a stress test that goes beyond the test code that was included in the starter files. In the later stages, we additionally introduce a coding style component. It is important to return any feedback right away so that it can be incorporated into the development of the next stage.

5 Further Extensions

There are multiple ways in which this assignment could be modified or extended.

- Collaboration: Instructors could assign this project in teams or require the use of pair programming.
- Assignment Scope: Instructors could assign only the first three parts (culminating in a console-based game) or only the final part (and provide the previous starter code to emphasize GUI aspects).
- Creativity: Students could be asked to design their own vehicles or to create additional levels themselves.
- Features: Students could incorporate additional features, such as a high score, timer, multiple levels, loading levels from files, and verifying level solvability.
- Data Structures: The assignment could also be extended to allow students to use different data structures (other than arrays).
- Autograders: Instructors could consider incorporating autograders for the assessment.

6 Materials

The following materials are provided:

- four numbered PDF files – assignment description for each phase of the project
- Rubrics.docx – rubrics for each of the project phases
- Interactivity Lab (optional).docx – an optional lab that provides practice on how to implement animations
- TrafficJamProject.zip – starter files for the Traffic Jam project, set up to be imported into Eclipse

Solutions can be provided upon request.

Acknowledgements

We thank the anonymous reviewers for their helpful comments.