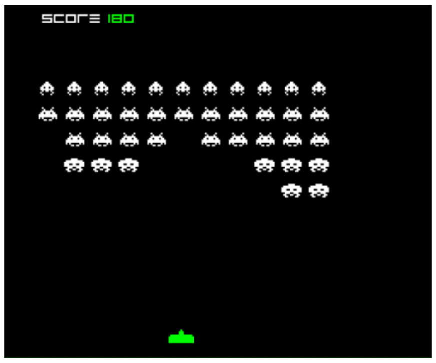
TUM Space Invaders (TSI) Problem Statement

1. The Problem

The market for computer games is moving towards complex, photo-realistic 3D games. However, such games have long startup times and high learning curves. They also require a lot of power and time.

Solitaire, Snake or Angry Birds are examples for the success of simple and easy to use games. They have a high fun factor, no learning curve and short startup times. As the duration of these games is short, they can be played in nearly every situation. TUM wants to have a simple and easy Space Invaders game and offer it for students for boring lectures. See the exemplary user interface on the right to get a feeling how the app should look like.



2. Scenarios

At the start of the TSI app, Alice, the player, has the chance to type in their player name. After typing in her name, Alice can see a menu, which offers 3 options: “Start”, “High scores” and “Quit”. Alice can navigate through these 3 options with the key arrows on her keyboard and can select one option by clicking the space key.
After selecting “Start”, Alice selects the 4th level and can start playing the game. She steers the TUM space ship to the left and the right and shoots rockets in order to destroy invaders. In case a rocket hits an invader, it disappears and a crash sound plays. The goal of the game is to remove all invaders from the game board as fast as possible. After Alice has destroyed all invaders, the game displays the total time that she needed. Alice is able to submit her personal high score in order to compare with other players of the TSI application.
While the game is running, there is music, when pausing the game, the music stops. During the game, Alice can inspect the already elapsed time and see the status how many invaders she already destroyed.

3. Requirements

The following minimal functional requirements (FR) and nonfunctional requirements (NFR) have to be addressed in the project:

FR1: Start and pause a game
A user can start and pause a game, this can be done either via buttons provided by the user interface or by pressing specific keys on the keyboard.

FR2: Select Level and support different types of Invaders
There should be at least 5 levels with different levels of difficulty. Different levels have a different arrangement of invaders and can have different types of invaders.

FR3: Steering the space ship with the keyboard
The space ship is steered with the arrow keys of the keyboard.

FR4: Shoot at Invaders and destroy them
By hitting the space bar a user can shoot rockets towards the invaders. When a rocket hits an invader, the invader as well as the rocket will be removed from the game board.

FR5: Share high scores for each level
After removing all invaders from the game board, the time is presented and the user interface offers the possibility to share the time with other players.

NFR1: Usability
The app should be intuitive to use and the user interface should be easy to understand.

NFR2: Conformance to guidelines
The design of the app should conform to the general usability guidelines for desktop operating system.

NFR3: Target platform
The app has to be developed in Java for desktop operating systems.

Additional constraints:

- The version control system must be git
- The app must be playable offline

In addition to the requirements above, the developers can come up with new ideas how to enhance the game and improve the game experience.

4. Target Environment

The application should be demonstrated in Java.

5. Deliverables

- Requirements Analysis Document (RAD)
- System Design Document (SDD)
- Source code under version control including source code documentation

6. Client Acceptance Criteria

The app must demonstrate the following functionality: It shows a list with at least 2 levels that can be played. The player can select and play a level until all invaders are destroyed.