

REQUIREMENTS DOCUMENT

1. SPL ARCADE GAME MAKER

Arcade Game Maker (AGM) is a pedagogical SPL for games created by the Software Engineering Institute (SEI) to support learning and experimenting with SPL concepts. It has a complete set of UML documents and models, as well as a set of tested classes and source code for three different games, namely: Pong, Bowling and Brickles.

Although not a commercial SPL, AGM has been used to illustrate the concepts of several distinct approaches to SPL and case studies and evaluation of SPL architectures. Its essential artifacts are: the characteristic model, the use-case model, the class model and the logical architecture of components, the latter will be presented in this document.

1.1. Similarities and Variabilities

The main similarities contained in the SPL AGM are presented as follow:

- every game has a set of Sprites, which are game elements that players see and interact with;
- every game has a set of Rules, which are the rules that govern the actions of the games. For example, a game might have a rule that a moving object when colliding with a static object must obey the laws of physics; and
- all games involve movement.

In addition to the similarities, SPL AGM has the following variability:

- Types of Rules: and the biggest difference between games. Some rules are related to the laws of physics (gravity, collisions, etc.) and may apply to multiple games. Other rules are game-specific and can be used in all game implementations, but do not apply to other games; and
- Types of Movement: in some games, movement is inherent to the game's operation. This happens periodically and is time-oriented. In other games, the player chooses and starts the movement, with actions being directed by the actor.

1.2. Actors and use cases

The following items present the actors and use cases of the SPL AGM and their main actions. Figure 3.1 presents the SPL AGM use-case diagram.

1. GamePlayer: actor responsible for performing the main actions of a game produced by AGM.
2. GameInstaller : actor responsible for game configuration actions (installation and uninstallation) produced by AGM.
3. Check Previous Best Score: verifies and presents the best score previously recorded.

- a. Actor selects the CHECK PREVIOUS BEST SCORE option from the system menu. System asks to provide the file name, reads the file and returns the score to the dialog box.
 - b. Actor selects OK in the dialog to continue. System returns to previous state.
4. Save Score: saves the player's current score.
 - a. Actor selects SAVE SCORE from the system menu. The system asks for a filename (creates a new one if it doesn't exist), writes the score to the file and returns to the pre-saved state of the game.
5. Save Game: saves the game in progress.
 - a. Actor selects the SAVE GAME option from the system menu. The system allows the actor to specify a filename and then writes the game data to the specified file, returning to the game's pre-saved state.
6. Game: installs the chosen game.
 - a. Actor chooses the game's installer to run. System presents a dialog box to choose the directory where the game files will be stored.
 - b. Actor chooses the directory. The system stores the files in the chosen directory.
- Exit Game: ends the game in progress.
- c. Actor selects EXIT in the System Menu. System presents dialog box for saving or exiting the game.
- d. Actor saves the game. System saves and exits or cancels game exit.
- e. System returns suspended action.
- Uninstall
7. Game: removes the selected game.
 - a. Actor chooses the UNINSTALL option from the system menu. System presents a dialog to the actor.
 - b. Actor selects the game directory to be removed. System deletes files from the directory and presents a complete removal dialog.
 - c. Actor selects the OK option from the dialog box. System closes the dialog box.
8. Play Selected Game: an actor selects the game and starts playing it.
 - a. Actor selects the PLAY option from the menu. System starts the game and presents the GameBoard.
 - b. Actor clicks the left button and starts the game. System starts game action.
 - c. Actor left clicks or uses keyboard to send commands. System responds to commands.
 - d. Actor responds to the Won/Lost/Even dialog by left clicking. System returns GameBoard to be initialized.
 - e. At any time the actor can select EXIT via the menu.
9. Play Bowling: starts the Bowling game.
 - a. Actor selects PLAY from the System Menu. System starts the game and presents the GameBoard.
 - b. Actor left click to play. System starts game action.
 - c. Actor repeats the following actions ten (10) times plus a bonus throw (optional)
 - d. Actor positions the mouse and clicks with the left button to throw the Bowling-Ball down the Alley (track). System moves BowlingBall through Alley using a

random algorithm. If there are collisions with BowlingPins, the system moves the BowlingPins according to the physical laws of collision. System counts the number of pins dropped. System computes the score.

10. Play Brickles: starts the Brickles game.

- a. Actor selects PLAY from the system menu. System starts the game and presents the GameBoard.
- b. Actor left click to start the game. System starts game action.
- c. Actor uses the left button or keyboard to send commands to the game. System moves the paddle horizontally, following the mouse trail. At each Puck movement, the system checks if there was a collision with another object. If the Puck collides with the Ceiling (ceiling) or a Wall, the Puck returns to the play area. If Puck collides with Floor, it ceases to exist. If the maximum number of Pucks has not been reached, a new one is created, otherwise the Lost dialog is displayed. If the Puck collides with a Brick, the action to be taken depends on the Brick. If it is the last Brick, the Won dialog is displayed.
- d. Actor responds to the Won/Lost dialog by left clicking. System returns the GameBoard to its initialized and ready-to-play state.

11. Play Pong: starts the Pong game.

- a. Actor selects PLAY from the system menu. System starts the game and presents the GameBoard.
- b. Actor left click to start the game. System starts game action.

12. Animation Loop: performs the animation actions of the games.

- a. System periodically generates signals and sends them to the game.
- b. System moves all objects step by step according to its moving algorithms.
- c. System checks for collisions and runs object collision algorithms.

13. Initialization: initializes the selected game and presents the gameboard.

- a. System creates default instances for required classes.
- b. System enters READY state.