"Karel Adventures" Game

Educational practice

18.06.2024

Oleksandr Hnutov, Orest Rublevskiy
National University of Kyiv-Mohyla Academy
2024

Table of contents

List of figures

1. Formulation of the problem

The primary problem addressed by “Karel Adventures” is the need for a practical educational tool that aligns with the curriculum of the 1st course of the “Software Engineering” specialty at the National University of Kyiv-Mohyla Academy. The game is designed to bridge the gap between theoretical knowledge and practical application, offering students an engaging way to learn programming and software development concepts.

2. Problem analysis

To effectively tackle the problem, “Karel Adventures” incorporates various learning objectives and technical challenges. The game features high-quality 3D graphics, player improvement systems, an inventory, and an open-world environment with different biomes. Additionally, it includes in-game “programming” using simple commands, which allows players to interact with the game world through coding. The project requires careful planning, robust implementation, and thorough testing to ensure it meets educational goals and runs smoothly on multiple platforms. All the features are implemented using modern jMonkeyEngine 3D game engine

3. Program structure

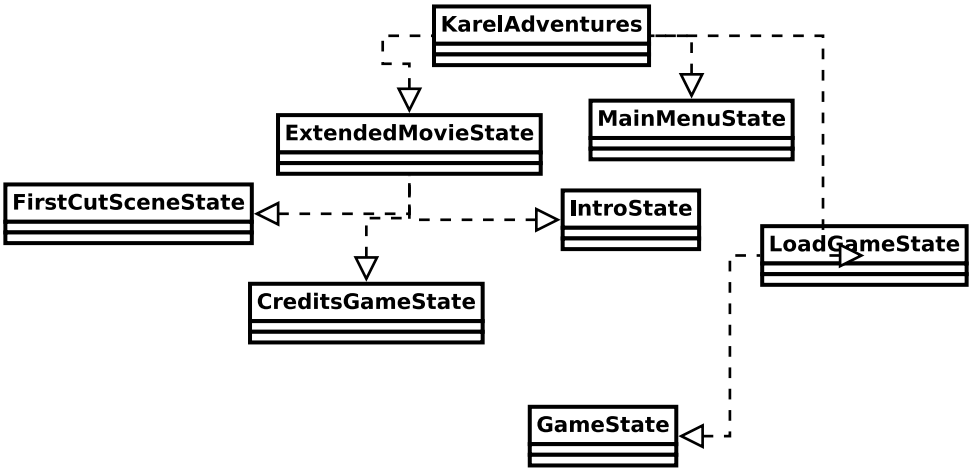

Figure 1 — Main program architecture

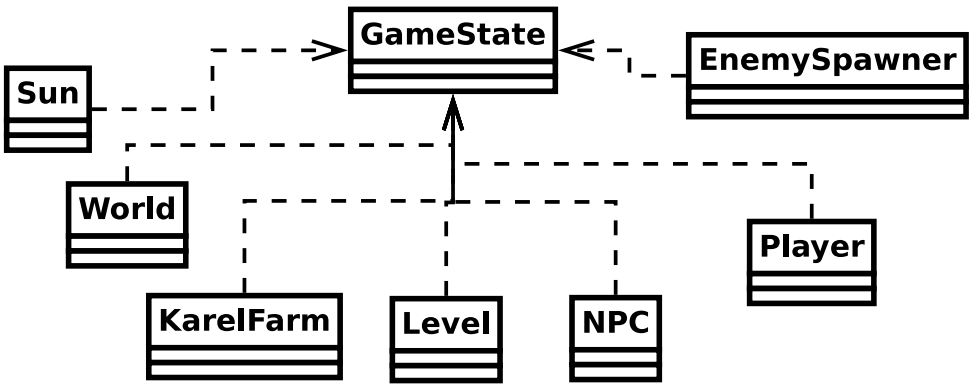

Figure 2 — Main game state relations

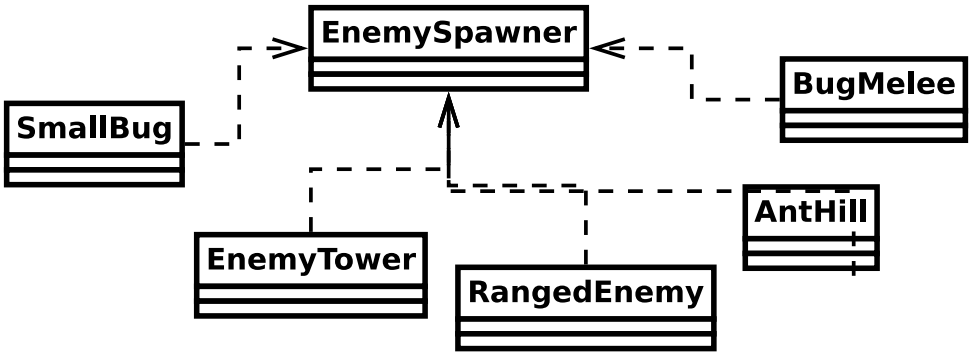

Figure 3 — Enemies system relations

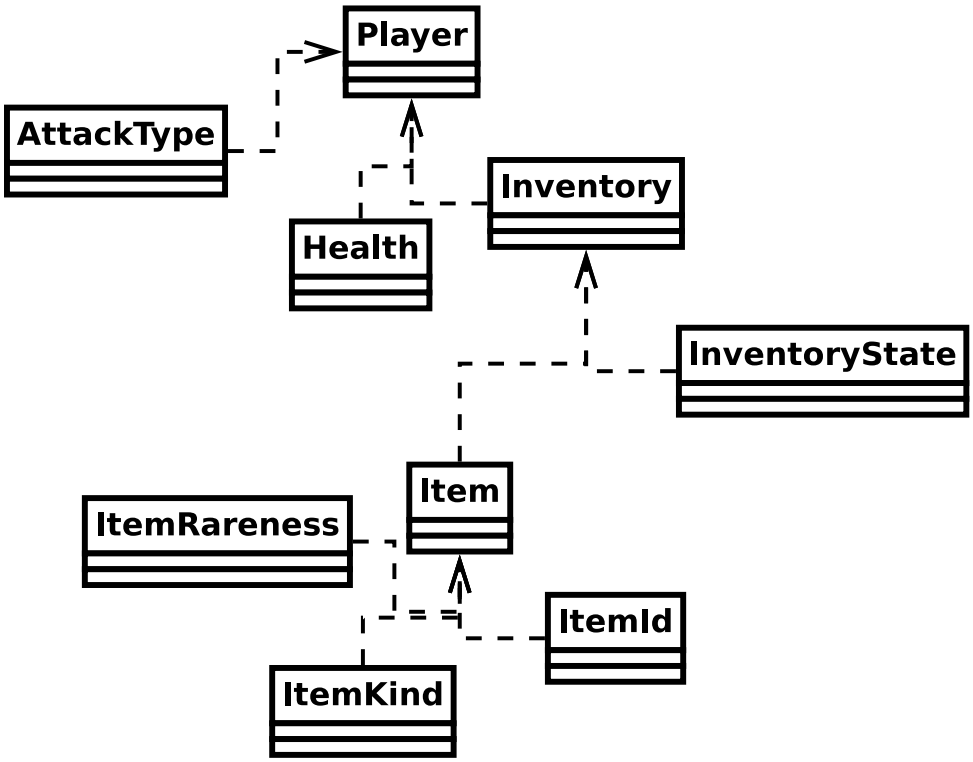

Figure 4 — Player system relations

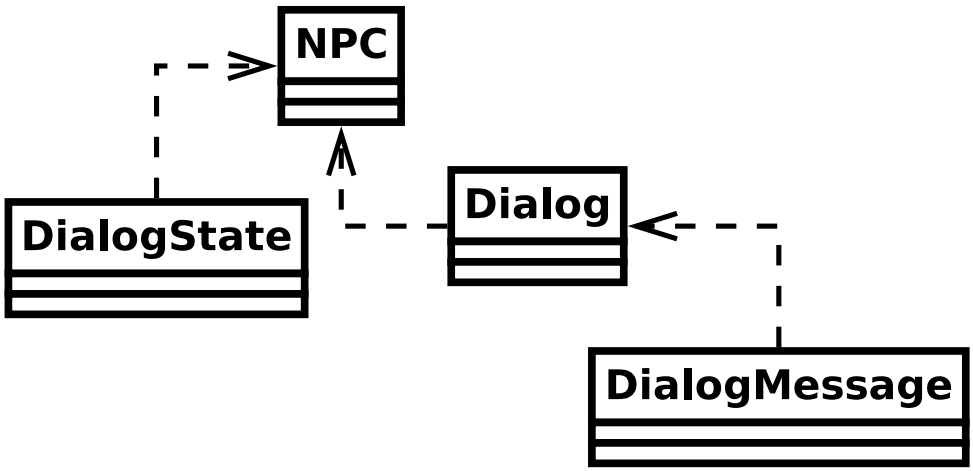

Figure 5 — NPC system relations

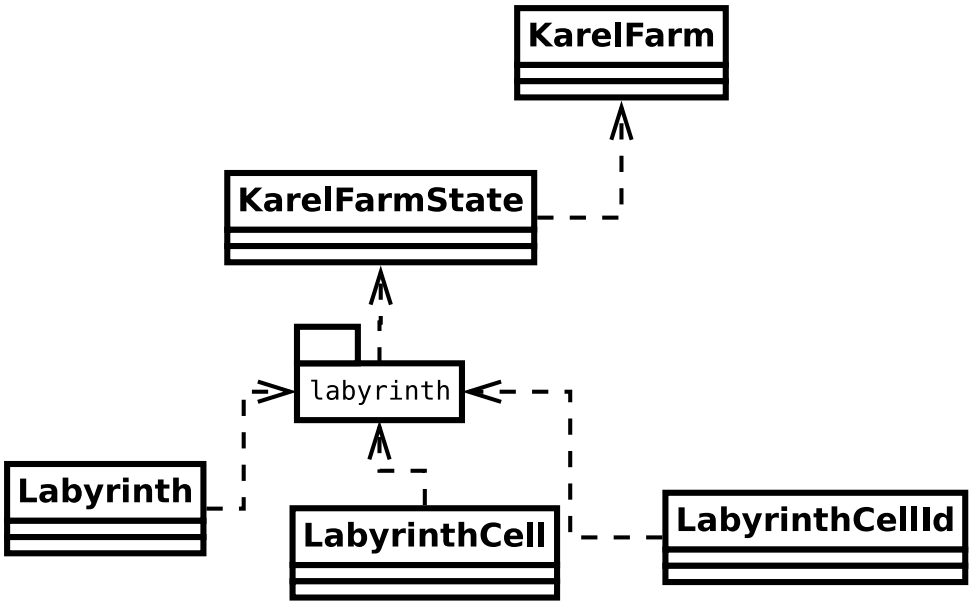

Figure 6 — Karel Farm farming system relations

4. Description of methods and classes

4.1. Core Game Loop

- `KarelAdventures.java` : This is the main class where the game starts and the main game loop runs. It initializes various components and manage the overall game flow.

4.2. Utils Package

This package contains utility classes for various functionalities:

- `TomlException.java` : Handles exceptions related to parsing TOML configuration files.
- `SaveLoadException.java` : Handles exceptions related to saving and loading game data.
- `ColorFormatter.java` : Deals with formatting and manipulating colors used in the console log.
- `InteractableNode.java` : Represents an interactive element within the game world.
- `AudioManager.java` : Manages sound effects and music in the game.
- `IAmEnemySpawner.java` : Interface defining methods for spawning enemies.
- `IUpdatable.java` : Interface defining an update method for objects that need to be updated each game tick.
- `Level.java` : Represents a level enumeration within the game.
- `SaveLoader.java` : Handles saving and loading game state.
- `IAmEnemy.java` : Interface defining methods for enemy behavior.
- `DebugLine.java` : Used for debugging purposes to draw lines for visualization.

4.3. State Package

This package contains classes representing different game states:

- `IntroState.java` : Handles the introduction sequence of the game (video).
- `DialogState.java` : Manages dialogue interactions with characters.
- `GuideBookState.java` : Represents the in-game guidebook state.

- `PauseState.java` : Handles the game paused state.
- `GameOverState.java` : Represents the game over state.
- `InventoryState.java` : Manages the player's inventory.
- `KarelFarmState.java` : Represents the Karel farm state related to a farming mechanic.
- `ExtendedMovieState.java` : Handles cutscenes and in-game movies (video).
- `GameState.java` : The core gameplay state.
- `LoadGameState.java` : Handles loading a saved game state.
- `MainMenuState.java` : Manages the main menu of the game.
- `FirstCutSceneState.java` : Handles the first cutscene of the game (video).
- `CreditsGameState.java` : Handles credits (video).

4.4. Game Package

This package contains core game logic and objects:

- `Sun.java` : Represents the sun object in the game world for lighting effects and post-processing.
- `World.java` : Represents the entire game world containing environment and objects.

4.5. Inventory Package

This package manages items and the player's inventory:

- `ItemKind.java` : Defines different types of items.
- `ItemRareness.java` : Defines the rarity of items (common, rare, etc.).
- `Inventory.java` : Represents the player's inventory.
- `ItemId.java` : Unique identifiers for items.
- `InvalidItemIdException.java` : Handles exceptions related to invalid item IDs.
- `Item.java` : Base class for all items in the game.

4.6. Player Package

This package defines the player character:

- `AttackType.java` : Defines different attack types the player can use.
- `Player.java` : Represents the player character with attributes like health and inventory.
- `Health.java` : Manages the player's health points.

4.7. NPC Package

This package deals with non-player characters (NPCs):

- `NPC.java` : Base class for all NPCs in the game.
- `DialogMessage.java` : Represents a single message in an NPC dialogue.
- `Dialog.java` : Manages NPC dialogues.

4.8. Enemies Package

This package defines various enemy types and their behaviors:

- `SmallBug.java` : Represents a small bug enemy.
- `BulletCollisionListener.java` : Handles collisions between bullets and enemies.
- `EnemyTower.java` : Represents an enemy tower that spawn bullets.
- `RangedEnemy.java` : Represents an enemy that attacks from a distance.

- `AntHill.java` : Represents an anthill that spawns ant enemies.
- `Bullet.java` : Represents a projectile fired by the player or enemies.
- `EnemySpawner.java` : Spawns enemies based on specific rules.
- `Boss.java` : Represents a boss enemy with unique behavior.
- `BugMelee.java` : Represents a bug enemy that attacks in melee range.

4.9. Farm Package

This package is related to a farming mechanic

- `KarelFarm.java` : Represents Karel Farm object used for items farming

4.10. UI Package

This package contains elements for the game's user interface:

- `PauseBlur.java` : Creates a blurred effect behind the pause menu.
- `Margin.java` : Defines margins for UI elements.
- `InterfaceBlurjava` : Creates a blurred effect behind a specific UI element.
- `labyrinth/**` : This subpackage deals with a labyrinth UI elements:
  - `LabyrinthCell.java` : Represents a cell within the labyrinth UI.
  - `Labyrinth.java` : Represents the entire labyrinth UI.
  - `KarelDirection.java` : Defines the directions for Karel's movement within the labyrinth.
  - `LabyrinthCellId.java` : Unique identifiers for cells within the labyrinth UI.
- `inventory/**` : This subpackage deals with the inventory UI:
  - `InventoryCellId.java` : Unique identifiers for cells within the inventory UI.
  - `InventoryCell.java` : Represents a cell within the inventory UI.
  - `InventoryHint.java` : Provides inventory items hint effect within UI
- `LoadingBarBuilder.java` : Creates and manages loading bars for the UI.
- `GuideBookTip.java` : Provides tips components for the guidebook UI.
- `InvalidCellIdException.java` : Handles exceptions related to invalid inventory cell IDs within UI
- `Logo.java` : Represents the game logo displayed in the UI.
- `ImageButton.java` : Represents a clickable image button in the UI.
- `FadePanel.java` : Controls the fading in and out of UI panels.
- `MsgBox.java` : Represents a message box element in the UI for displaying messages or dialogs.

5. Distribution of roles

Oleksandr Hnutov	Orest Rublevskiy
User interface	Player movement
Project architecture	Enemies logic
Sound effects	Animation
Music	Health/regen system
Level design	Dialogs
Reports	Plot
Karel Farm	

Table 1 — Distribution of roles

6. Problems during development

1. **Leveraging Jmonkeyengine for 2D/Labyrinth Elements:**

We implemented a "Karel Farm" mechanic that utilizes a labyrinth and basic coding commands. jMonkeyEngine is primarily focused on 3D game development. While it's possible to create 2D elements within Nifty GUI library, it might not be the most efficient approach. This could lead to:

- *Workarounds and Hacks*: Developers might need to implement workarounds or hacks to achieve a smooth 2D experience within a UI framework and 3D engine.
- *Performance Issues*: Rendering 2D elements within a 3D engine might be less performant compared to using a dedicated 2D game engine.

2. **Balancing Difficulty and Educational Value**:

The game aims to teach programming concepts while offering an engaging horror RPG experience. Balancing these aspects can be tricky:

- **Making it too easy**: If the coding challenges in the "Karel Farm" are too simple, they might not provide a strong learning experience.
- **Making it too hard**: Overly difficult coding challenges could frustrate players and hinder their enjoyment of the game.

3. **Debugging and Optimization for Multiple Platforms:**

jMonkeyEngine supports multiple platforms (Windows, macOS, Linux). However, debugging and optimizing the game for each platform can be time-consuming:

- **Platform-specific Issues**: Different platforms might have their own quirks and require specific optimizations.
- **Testing Challenges**: Thoroughly testing the game across all platforms to ensure a smooth experience can be a significant undertaking.

4. **Resource Management and 3D Open World:**

The text describes a 3D open world with various biomes. Creating such an environment requires careful resource management:

- **Memory and Performance**: A large open world with complex assets can strain memory and processing power, impacting performance.
- **Level Design and Content Creation**: Filling a vast open world with engaging content and maintaining a consistent visual style can be a significant workload for developers.

7. User manual with illustrations

1. You are playing as a Karel - robot, which goal is to save his computer world from **bugs**
2. In order to complete the game, you must interact with NPCs, kill enemies, earn money and play Karel Farm farming mechanism to get more exclusive armor and weapon. Your main goal is to kill the final boss.
3. Main screens and interactions with them:

Figure 7 — Main menu screen

Here you can choose to start a new game, load saving (if exists) or exit to OS.

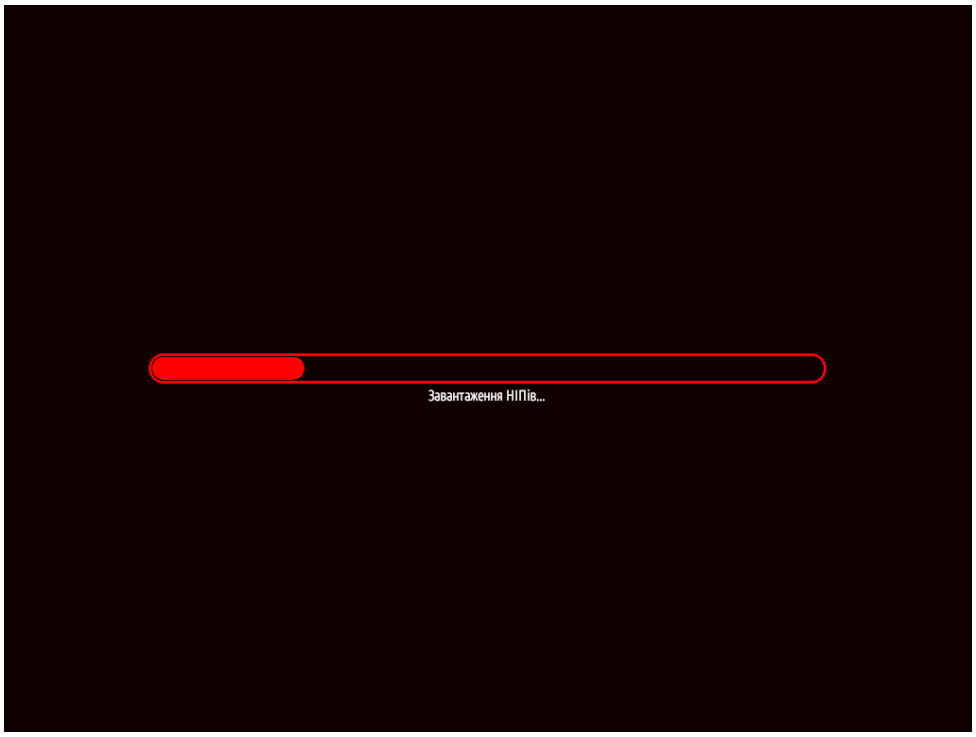

Figure 8 — Loading screen

May not be skipped and may take a while to load everything needed.

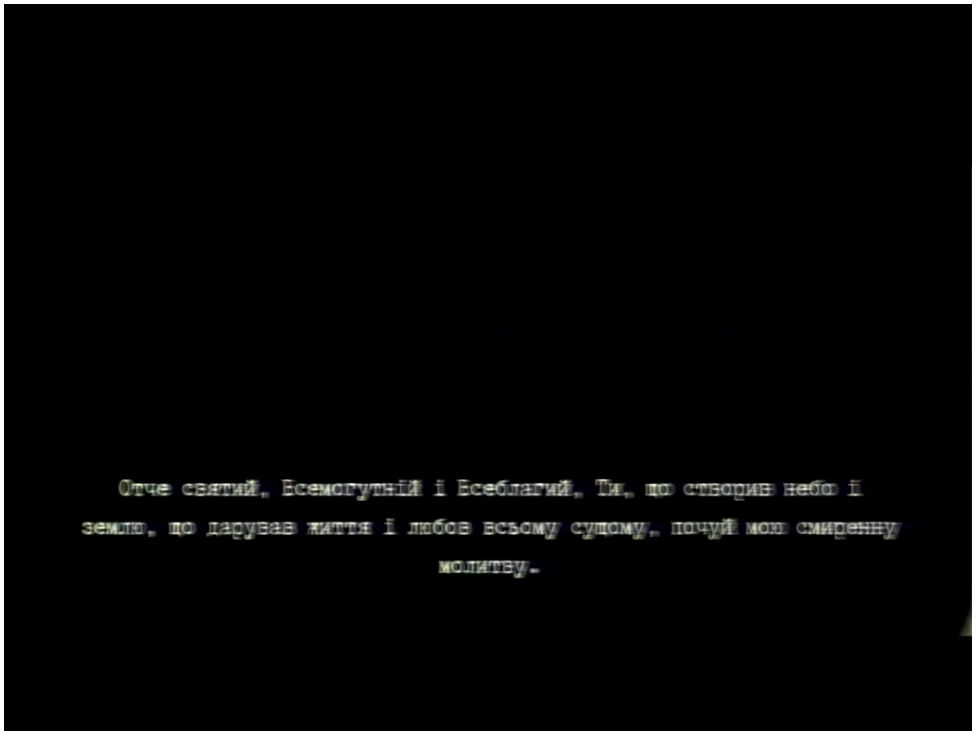

Figure 9 — First cutscene

It partially explains the game plot. May be skipped.

Figure 10 — Main game state screen

There you can see stats (e.g. health and balance), character icon and guidebook, which can be accessed with **G** key

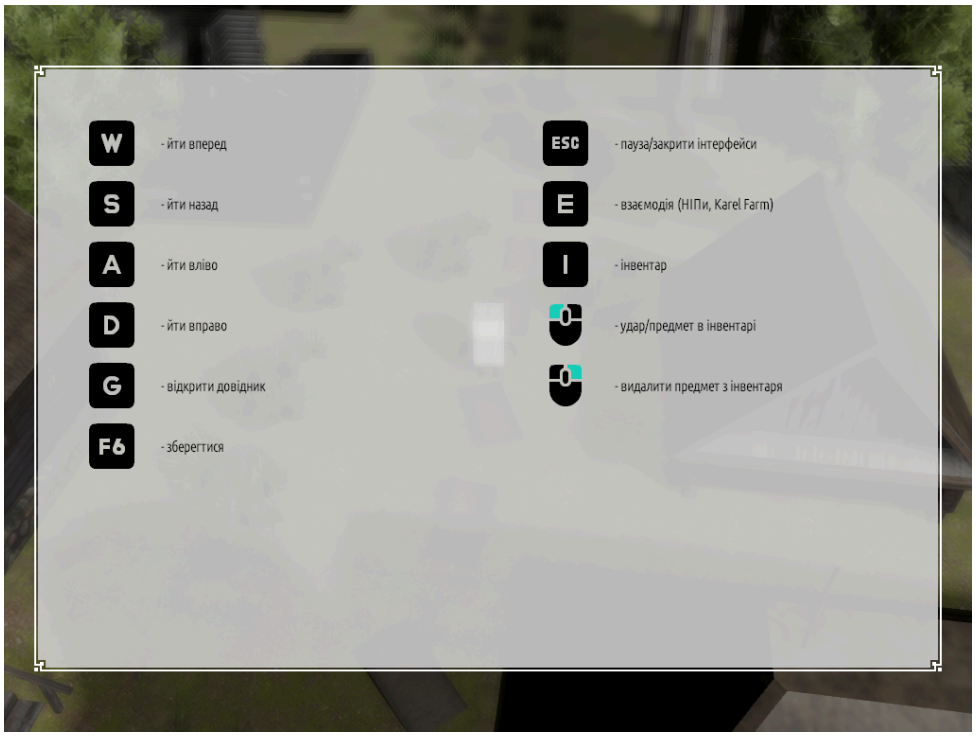

Figure 11 — Guidebook screen

It contains all necessary keyboard keys to interact with the game

Figure 12 — Karel Farm

It is a farming mechanism, where you can spend your money to pass the labyrinth and collect all the beepers to get a random item. Some labyrinths may be unpassable (Karel or beepers may be locked in walls). If so you must rebuild the labyrinth, which also costs money. To pass the labyrinth, you must enter commands to move Karel. If a wrong command is printed, available commands list is displayed.

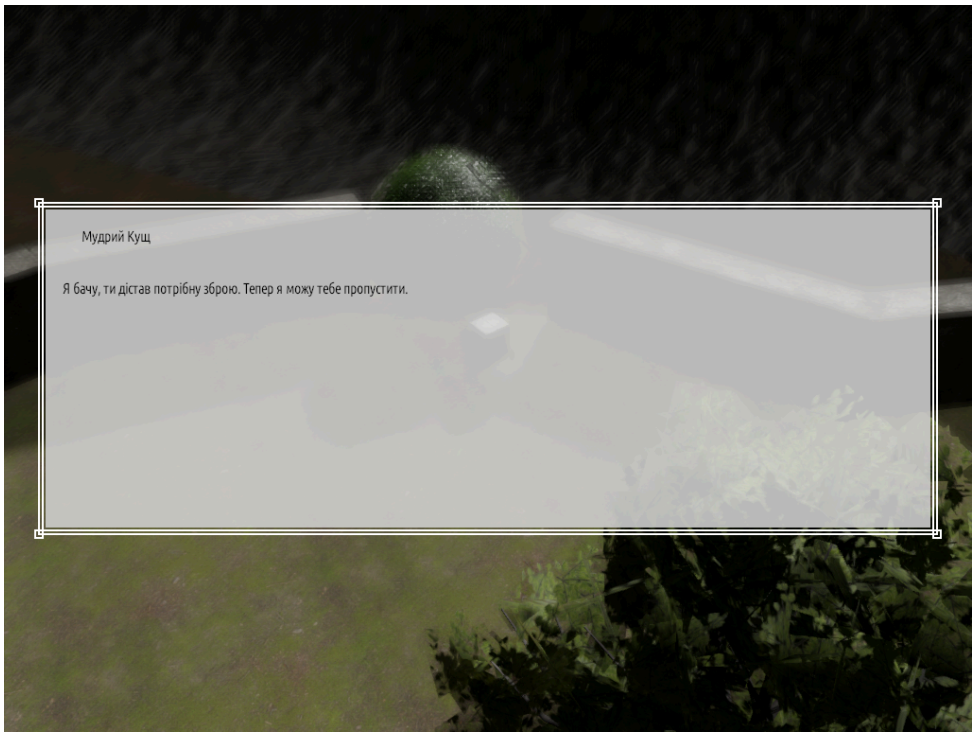

Figure 13 — Dialog screen

You must press **Return** to continue the dialog, or **Escape** to exit.

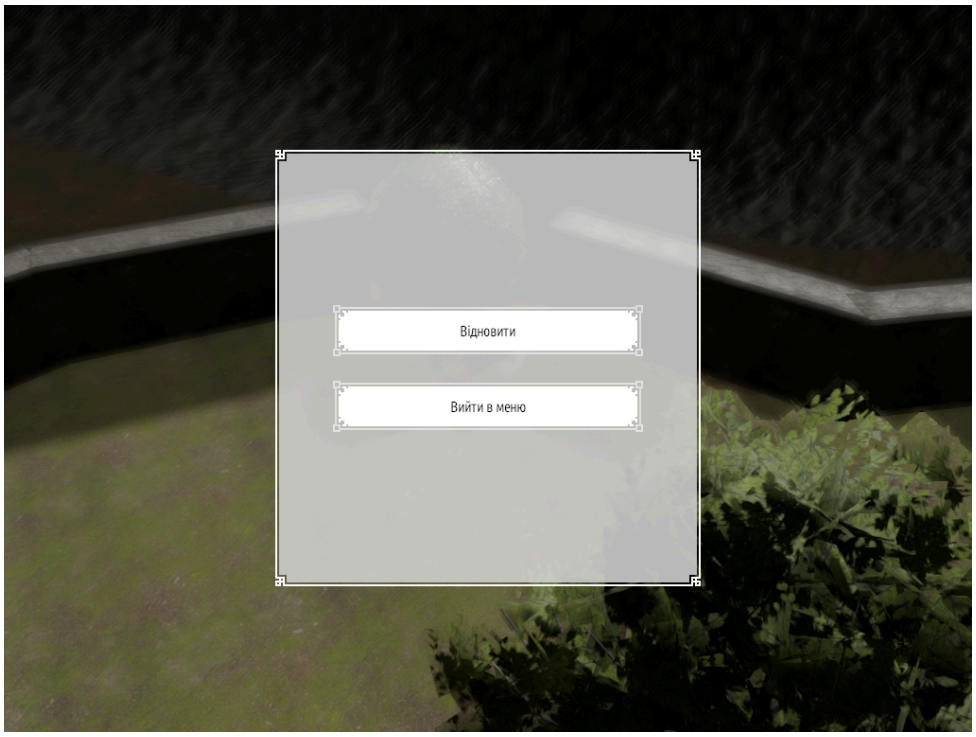

Figure 14 — Pause menu

Here you can exit to menu or resume the game

Figure 15 — Inventory screen

You can move items to different cells in order to equip/un-equip armor/weapon, right-click to remove items and hover them to discover their stats (name, description and benefit). Also color of the stats displays their **rareness**:

- Black is Common

- Green is Rare
- Grey is Silver
- Yellow is Golden
- Magenta is Legendary

8. Conclusion

"Karel Adventures" presents itself as a unique and engaging approach to learning programming concepts. By combining horror RPG elements with a 3D open world and a "Karel Farm" mechanic that utilizes basic coding commands, the game offers a fun and interactive way for students to grasp essential programming principles. The detailed class breakdown showcases the project's well-structured architecture, and the distribution of roles highlights the collaborative effort behind its development.

This educational game has the potential to:

- Motivate students by gamifying the learning process.
- Bridge the gap between theoretical knowledge and practical application.
- Provide a platform for students to experiment with coding concepts in a safe environment.
- Enhance engagement with software development principles.

"Karel Adventures" holds promise as a valuable tool for educators seeking to introduce programming concepts in a novel and captivating way.

9. Software code listing

https://github.com/konceptosociala/KarelAdventures