Snake

Team HHC

Li Cao licao@bu.edu

Yucheng He hyc1011@bu.edu

Lixu He masonhe@bu.edu

Github Link: https://github.com/licao2998/535-project
YouTube: https://www.youtube.com/watch?v=mLEJlaM-pgw

Abstract

In this project report, we present a novel implementation of the classic Snake game on the Beagleboard platform, with three different difficulty levels (easy, medium, and hard) and three different maps to cater to a wide range of player abilities needs. A key innovation of our version is the inclusion of an AI Snake, enabling players to challenge an artificial intelligence opponent in a dynamic and engaging gameplay experience. This enhanced Snake game leverages the computational capabilities of the Beagleboard to deliver a responsive and visually appealing interface, while the AI Snake demonstrates the application of artificial intelligence in traditional gaming scenarios. The report details the design, development, and performance evaluation of the game, illustrating the effectiveness of our implementation in providing an entertaining and challenging experience for players of all skill levels.

1. Introduction

The project topic at the heart of this report is the development and implementation of an innovative version of the classic Snake game on the Beagleboard platform. The primary objective is to modernize the beloved game by incorporating multiple levels of difficulty and a competitive AI opponent while maintaining its nostalgic charm. In doing so, we aim to create a gaming experience that caters to a wide array of players, regardless of their skill set or familiarity with the original game. Moreover, by integrating artificial intelligence and advanced graphics, we strive to demonstrate the potential of the Beagleboard platform for developing engaging and challenging games.

The "big picture" overview of the project components can be broken down into several key areas. Firstly, the design and development of the game interface have been a major focus, as it plays a critical role in delivering an engaging and enjoyable experience for players. This includes optimizing the graphics and user interface for the Beagleboard platform while ensuring that the controls remain intuitive and user-friendly.

Secondly, the artificial intelligence snake integrated with artificial intelligence capabilities is also an important part of the project. We developed an artificial intelligence opponent. In the game, we not only need to compete with the AI snake for food but also need to pay attention to avoiding the AI snake. The AI snake, both as a competitor for food and as a constantly moving obstacle, not only enriches the gameplay but is also an example of how artificial intelligence can be successfully integrated into traditional game scenarios.

Thirdly, the creation of distinct difficulty levels has been instrumental in ensuring that the game is accessible to a broad range of players. The three levels—simple, medium, and difficult—allow users to tailor their gameplay experience according to their skill level, providing a more personalized and satisfying gaming experience.

Furthermore, the project highlights the effective utilization of the Beagleboard's computational power to deliver a smooth and visually appealing gaming experience. By optimizing the game's performance on the platform, we have been able to ensure seamless and responsive gameplay that meets the high standards set by modern gaming experiences.

As a result of our diligent efforts, we have successfully implemented an enhanced version of the Snake game that offers three levels of difficulty and features a competitive AI Snake. Our implementation demonstrates the seamless integration of artificial intelligence within the context of traditional gaming, providing a stimulating and entertaining experience for players across all skill levels. Additionally, this project serves as a testament to the versatility and potential of the Beagleboard platform for game development, as well as the value of incorporating AI technologies in creating engaging and dynamic

gaming experiences. In conclusion, we believe that our innovative approach to revamping the classic Snake game showcases the immense potential for blending nostalgia with modern advancements in the gaming industry.

2. Design Flow

In this project, we have developed an enhanced version of the classic Snake game on the Beagleboard platform. Our implementation consists of several interconnected components that work in harmony to provide an engaging and enjoyable gaming experience. These components are User Interface, Game Logic, and Graphics.

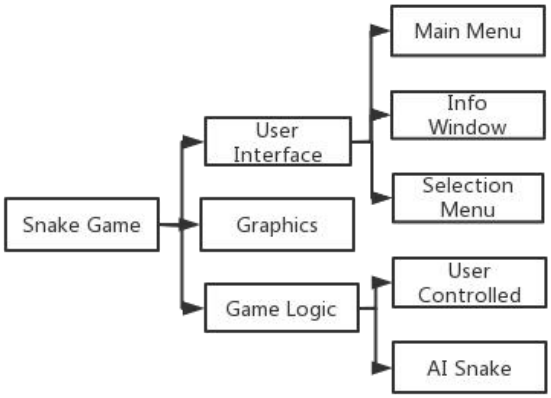

Figure1. System architecture

The User Interface component is the primary medium for players to interact with the game, receiving input from players and providing visual feedback on the game's state, performance, and changes. The Game Logic Module is the core of the game, implementing rules mechanics, managing the game state, coordinating other activities, and controlling the AI Snake. The various components of the project interconnect to create a seamless and engaging gaming experience, showcasing the potential of the Beagleboard platform and demonstrating the integration of artificial intelligence in traditional gaming scenarios.

In conclusion, the various components of our project interconnect and collaborate to create an enhanced Snake game that offers a challenging and entertaining experience to players of all skill levels. By carefully designing and implementing each module, we have ensured that the game functions seamlessly and provides a visually appealing, responsive, and enjoyable experience for the players. Our project demonstrates the potential of the Beagleboard platform for game development and serves as an example of how artificial intelligence can be integrated into traditional gaming scenarios to create a more dynamic and engaging experience.

Function	Member
Mainwindow setup & buttons setup	Lixu He
Map&level choices	Lixu He
Rank	Li Cao
Food setup & refresh	Li Cao
Snake move & collide detection	Yucheng He
Game logic	Yucheng He
AI snake move	Li Cao & Yucheng He

Maps setup & "Game Over" infowidget	Li Cao & Yucheng He
Report	Li Cao & Yucheng He & Lixu He

Percentage:
- Yucheng He 35%, Licao 35% and Lixu He 30%

3. Project Details

a. User Interface Module

The User Interface Module forms the backbone of our game's presentation and player interaction. We began by conducting preliminary research on existing Snake game implementations, examining their design choices and usability [1]. Based on our findings, we designed a clean, intuitive, and visually appealing interface that allows players to easily navigate menus, adjust settings, and engage with the game.

We utilized the QT library [2] for creating the game window, handling user input, and rendering graphics. We implemented custom fonts and color schemes to enhance the game's visual appeal and improve readability. We also included a separate screen for displaying the player's score, game progress, and other relevant statistics.

In our project, mainWindow.cpp is the relevant code for the main interface. We have set different difficulty gradients here, allowing players to interact with the game through button events.

Figure2. The initial UI of the game.

b. Game Logic Module

First initialize the game by setting up the UI, creating the walls, initializing the snake, and connecting the buttons to their respective functions. A timer is started when the game is started and the “timeOut” function is called every time the timer times out.

The “timeOut” function is the heart of the game logic. It checks whether the snake has collided with itself or the wall by calling “snakeStrike” which first loops through each segment of the snake's body and then loops through all subsequent segments using a nested loop for each segment. If the coordinates of the two segments are the same, the snake has collided with itself. If there is a collision, the game is over and an pop-up window is displayed.

The snake's movement is based on the four direction buttons. When the player clicks on one of these buttons, a corresponding direction flag is set. The movement of the snake is then determined based on this flag. There are four corresponding functions. Each of these functions

first checks whether the new position of the snake's head will result in a collision with the walls or with itself. Then each of them inserts a new rectangle at the front of the snake's body, with the point of the rectangle shifted. For example, if the up button is clicked, the “addTopRectF” function is called and it inserts a new rectangle at the front of the snake's body, with the top left point of the rectangle shifted upward by the height of one rectangle. If the top of the snake's head is already at the top of the screen, a new rectangle is inserted at the bottom of the screen. After the insertion, The deleteLastRectF() function is called to remove the last rectangle in the snake's body to keep the length of the snake constant.

In this module, the food is randomly generated on the board when the game starts and is refreshed after the snake eats the previous food. The “x_notin_block” function is used to generate a random point on the board that is not already occupied by the snake or the wall by checking if the coordinate for the food point is occupied by the snake or wall.

When the snake's head collides with the food, the “timeOut” function detects this collision by checking if the head of the snake intersects with the food. If there is a collision, the “addscore” function is called to increment the score by 5, the food is removed from the board, and the “rewardTimeOut” function is called to generates a new food point by appending it to the rewardNode list until there are at least two food points on the board. The function also adds a new block to the snake's body based on the direction of the snake's movement to make it grow. Finally, the update() function is called to update all new changes.

When the start button is pressed, the timer is either stopped or started, and the text of the stop button is updated accordingly. The “end” button and “save” button are connected to functions that stop the game and also allow the player to save their game data or delete it if the game has ended.

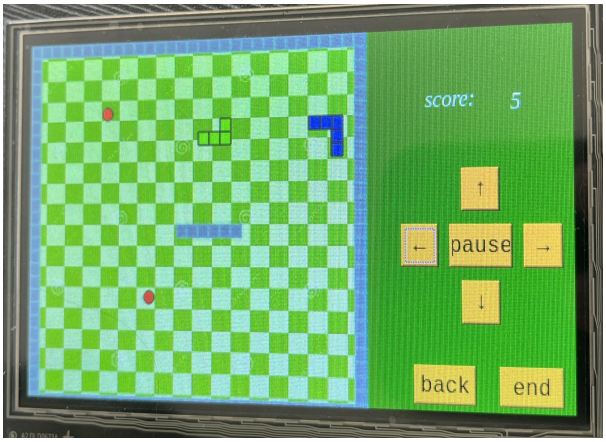

Figure3. The gaming interface.

c. AI snake module

The AI snake's movement direction is randomly selected using the std::rand() function, and a corresponding function is called to move the AI snake in that direction. These four movement functions (ai_addRightRectF(), ai_addLeftRectF(), ai_addTopRectF(), and ai_addDownRectF()) are used to simulate the Snake's movement. They move the AI snake by inserting a new rectangle representing the Snake's head at the beginning of the Snake's list of rectangles while deleting the last rectangle in the list.

To move the AI snake, a direction variable (ai_moveFlage) is set based on the chosen direction, and a corresponding movement function is called. This movement functions to move the AI snake in the specified direction by calculating the new position of the Snake's head and checking for collisions with walls or other objects in the game. If a collision is detected, the AI snake's movement is not executed, and the function does nothing. The specific function called depends on the chosen direction of movement.

The timeOut() function repeatedly calls aiSnake() to update the movement of the AI snake. It checks if the AI snake collides with any rewards or itself, removing the reward and calling aiSnake() again to continue moving the AI snake if necessary. The function also generates new rewards if there are fewer than two in the game.

d. Save game status module

The Game Data Module stores game data, such as snake position, level, score, map, direction, and food position, enabling players to resume the game later. Snake position is captured by looping through the "snake" vector, retrieving x and y coordinates with the "topLeft()" function, and storing them in a QJsonObject added to a QJsonArray holding all game data. All other data is handled in a similar way. The QJsonArray is then converted to a QByteArray using "toJson()," which is saved to "game. data" with the "write()" function, saving the game state.

To restore the game state, JSON data is read from "game. data" and converted to a QByteArray, which creates a QJsonDocument object. The QJsonDocument object extracts game data items to restore the game state.

e . Rank module

To implement a ranking function so that the player can check the ranking status in the main interface, we need to save the score of a finished game to a file and have all the scores in descending order. If the file does not exist, the player's score is simply added to the file. If the file already exists, first read the scores from the file, append the current score to the end, and then sort and save the scores.

In this module, the player's score is saved to a file called "rank. data". If the file exists, it reads the scores, adds the current score to the list, and sorts the list in descending order by the combined use of the "std::sort" function, which is a standard C++ function that sorts a range of elements in ascending order, and a lambda function to reverse the sort order. Then the sorted scores are written back to the file. If the file doesn't exist, it creates a new file and writes the current score to it.

To write the sorted scores back, the module erases the contents of the file using the "resize()" function with a size of 0 and uses the "write()" function after converting scores to strings with newline characters.

f. Difficulties in debugging

First of all, in the process of code writing, the code logic of this project not only requires people to control the direction of the Snake so that the Snake can accurately eat food to get points and

increase its length by one but also needs to set up an automatic running snake. The crawling direction and obstacle avoidance of this Snake requires code to try to write, so it is relatively difficult to add the logic part of the operating mode of the AI snake.

g.Future Improvement Direction

First of all, regarding the control algorithm of the direction of travel of the AI snake, this project has a lot of room for improvement. An efficient algorithm can be used in the code to calculate the shortest path for the AI snake to obtain food. Allowing AI snakes to compete for food with fast and accurate paths greatly improves the fun of the game and also greatly improves the competitiveness of the game. Second, this project can add audio modules. Different maps and difficulty levels can set different background music, and the background music with a stronger rhythm matches the higher difficulty level. Upbeat and rhythmic music will make users more engaged and passionate in the difficult mode of the game. Finally, this project can also add more input methods so that users can choose multiplayer mode or man-machine mode.

h. Documentation and Source Code Management

To facilitate collaboration and maintain a well-organized codebase, we used the GitHub as a central repository for our project. We also ensured that our source code was well-documented, including comments, function descriptions, and design explanations, to enhance readability and maintainability.

i. Commands and programming languages
Programming language used by the code: c++
Commands required for this project:
qmake
make
./snake

3. Summary

Our project developed an enhanced version of the classic Snake game on the Beagleboard platform with the goal of creating an engaging and challenging gaming experience. Our accomplishments include the successful design and development of an intuitive and visually appealing User Interface optimized for the Beagleboard platform, as well as the implementation of three difficulty levels catering to different player skill sets.

However, there remain several challenges and areas for improvement. One such challenge is enhancing the AI Snake's behavior to make it more adaptable and unpredictable. Additionally, exploring cross-platform compatibility and porting the game to other platforms, such as mobile devices or web browsers, could increase the game's reach and audience.

In summary, our project successfully demonstrates the potential of the Beagleboard platform for game development and showcases how artificial intelligence can be integrated into traditional gaming scenarios to create a more dynamic and engaging experience. While there are remaining challenges and areas for improvement, our accomplishments in developing an enhanced Snake game serve as a strong foundation for future development and expansion.

References

[1] Classic Snake Game Implementations. Various sources. Accessed from https://github.com/topics/snake-game
[2] QT Library. Qt | Cross-platform Software Design and Development Tools
[3] Liao, C., Zhang, Y., & Yang, J. (2019). Development and Research of Mobile Game Based on BeagleBone Black. In 2019 IEEE 3rd Information Technology, Networking, Electronic and Automation Control Conference (ITNEC) (pp. 2258-2262). IEEE.
[4] Kaur, M., & Rana, R. (2019). Design and implementation of Snake game using C++. In 2019 5th International Conference on Computing Sciences (ICCS) (pp. 21-26). IEEE.