UPPSALA UNIVERSITY

Group Project Report

Group 17

Authors:
Arnab Kumar Ghosh
Biruk aklilu
Julios Fotiou
Sotirios Aias Karioris

October 22, 2025

Abstract

This report presents *PiTris*, a simplified Tetris game developed on the Raspberry Pi with the sense HAT. The game uses the accelerometer for tilt-based controls and displays blocks on the 8 x 8 LED matrix. The final implementation confirmed that motion control is workable and responsive, but also highlighted challenges with calibration and limited display resolution. Through development and testing, we gained practical experience with integrating hardware with game logic and identified areas where visual clarity and input sensitivity could be improved. The project demonstrates the feasibility of interactive gameplay on compact hardware while revealing useful technical insights.

Contents

Chapter 1

Introduction

The project *PiTris* is our attempt to recreate the classic Tetris game on the Raspberry Pi. Motion-based interaction has become increasingly common in gaming and embedded systems because it creates an intuitive and engaging user experience . The sense 8 x 8 LED matrix of HAT provides visual output showing the blocks as they fall from the top of the screen,the sense HAT is frequently used in educational and experimental projects due to its compact design and integrated sensors, making it suitable for simple hardware-based games[1] . The players aim to align the blocks to form complete rows, which then disappear as in the original game.

This project is interesting because it combines software development with hardware interaction. We work on game logic, timing, and scoring while also interpreting sensor data, handling calibration, and managing the limited resolution of the LED display. In its final state, the project includes a fully functioning game with motion controls, refined through development and testing to ensure responsiveness and playability.

1.1 Purpose and Goals

The main purpose of this project is to gain practical experience in designing and building an interactive system that connects hardware with software. We aim to create a game that is fun and engaging while learning the challenges of working with real hardware. This project also helps us understand how users interact with devices and how input affects game performance. In addition, it gives us experience in planning, testing, and improving a system from

concept to working prototype.

Our specific goals include the following.

- Develop a version of Tetris that works on the Sense HAT's LED display, keeping in mind its small size and unique visual limits.
- Learn how to read accelerometer data and turn them into smooth and reliable player controls.
- Understand the challenges of hardware, including calibration, responsiveness, and software integration.
- Consider the wider aspects of the project, such as making it accessible to many users, promoting inclusivity, and thinking about ethical and environmental impacts.
- Explore ways to improve the user experience by testing different control schemes and game speeds.
- Document the development process thoroughly to help others learn from our approach and solutions.

1.2 Project Outline

This report is structured as follows.

- **Chapter 2** reviews background material and related projects using the Raspberry Pi and Sense HAT.
- **Chapter 3** explains the methods used for development, testing, and prototype design.
- **Chapter 4** presents the implementation so far, focusing on the accelerometer prototype and integration challenges.
- **Chapter 5** discusses the technical, organizational, and developmental challenges encountered during the project and explains how they were addressed to ensure successful implementation
- **Chapter 6** discusses ethical considerations,

- **Chapter 7** details the contributions of each member of the group.
- **Chapter 8** concludes with reflections on progress and describes future steps to complete the project.

Chapter 2

Background

The Raspberry Pi has become a popular platform for developing interactive and educational games due to its low cost, accessibility, and support for multiple programming environments. Previous studies have demonstrated its effectiveness in teaching computing concepts and enabling motion-based game prototypes, although challenges such as limited processing power, low-resolution displays, and delayed sensor input have been documented [2]. Research also shows that small-scale Raspberry Pi gaming systems can promote learning engagement, but their hardware constraints often require design trade-offs in responsiveness and graphics quality.

To expand its capabilities, accessories such as the Sense HAT provide additional components, including an 8×8 LED matrix, gyroscope, and accelerometer, which allow for sensor-based interaction. Prior work involving tilt- or motion-controlled games using the Sense HAT has highlighted both its potential for interactive applications and its limitations in terms of display size and sensor precision [3]. These findings suggest that compact hardware can support engaging gameplay experiences, provided that input handling and visual representation are carefully managed.

Building on these earlier efforts, the PiTris project implements a Tetris-style game that relies on tilt-based input using the Sense HAT. The game explores hardware–software integration, real-time sensor monitoring, and the constraints of minimal display output, while addressing technical challenges noted in previous research.

Chapter 3

Methodology

Before explaining the details of *PiTris*, it is important to understand how the project builds on the concepts discussed in the background. The tools, programs and programming language used in the project are listed in this chapter. Additionally, the way the source code constituting the program is described, thus showcasing how the general problem can be broken down into manageable tasks suitable for a team.

3.1 Programming Language and Tools Used

The source code for *PiTris* is exclusively written in Python, a language known for its flexibility and ease of use for rapid application development. For the utilization of the peripherals found in Sense HAT the appropriate API has been employed. Version control is provided by *git* while for the software's distribution and synchronization between the members *GitHub* has been chosen.

3.2 Division of Workload

Even though the target goal of this project is relatively straightforward, the work itself resembles typical game development since the game at hand can be essentially split logically into two parts: the *game logic* and the *control logic*. The *game logic* is the field of the source code that implements the actual game, such as its rules and its graphics. The *control logic* in this case revolves around the use of the input sensors of the hardware for their

implementation into the final game. These two facets of the program can be independently developed and then integrated later on.

The game logic side is more layered and complex compared to the control logic. Game logic encapsulates the software routines responsible for drawing to the LED grid, the control of the logical state the program is in at any given moment, and the processing of user input for the generation of actions within the game. Splitting all of these parts of the game logic into smaller parts, it is possible for different members to work simultaneously.

Chapter 4

Implementation

This chapter describes the program architecture and details about the source code.

4.1 Program Architecture

PiTris is implemented with a continuous while-loop and a finite state machine. The state machine contains three functions where each one corresponds to one of the basic states the game may be in: waiting for a game to start, playing a game and the game over screen. At the end of each loop iteration execution pauses for a few milliseconds, thus giving the player time to process what is happening in the game. The functions that will be executed in each iteration depend on the current state but all states will execute one function to draw pixels to the screen, one function to poll inputs and at least one function to determine what changes should occur in the internal sub-states of the game.

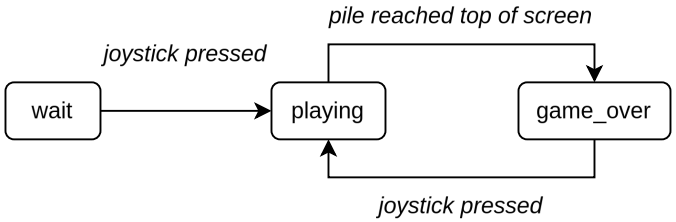

Figure 4.1: State Machine Flow Diagram

The Wait and Game Over functions are similar; they simply display text with the built-in `show_message()` command from the Sense HAT API. The playing state consists of the larger part of the code base, being responsible for handling the game physics and logic, as well as converting movement picked up by the accelerometer into inputs understandable by the game logic.

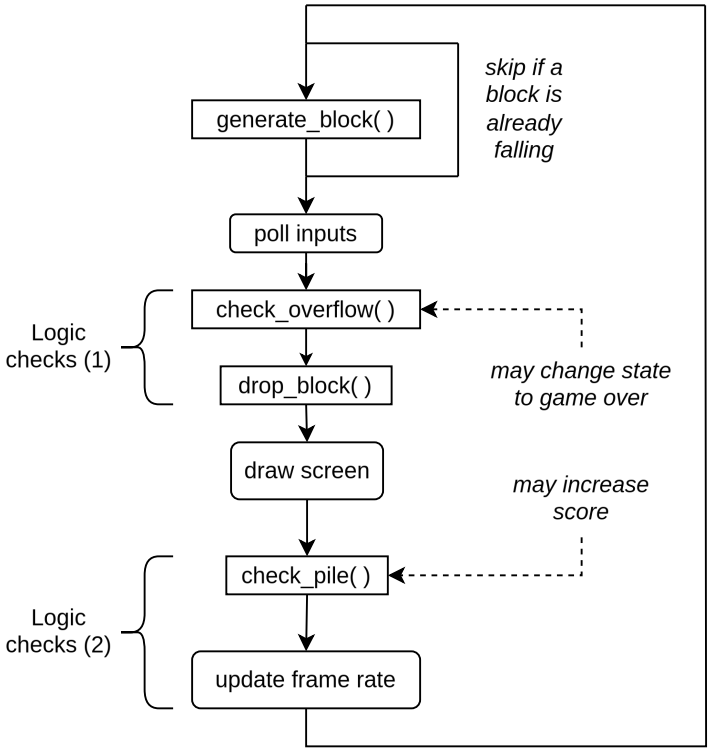

Figure 4.2: Skeleton of Playing State Loop Iteration

During playing, a single block will drop by one line every two frames, even though the screen will be drawn and inputs will be polled every frame. This ensures smooth and responsive gameplay. Once a block falls to the bottom of the screen or touches other fallen blocks it becomes part of the pile object. At the start of the loop, if no block is falling due to the previous one joining the pile in the last frame, the `generate_block()` function will create a new block, selecting randomly between 5 predetermined shapes. After polling the inputs, the game will execute a small set of functions to determine if the next state will be game over and where the falling block should be placed, based on current input and pile state. Then, after drawing the new frame

on screen, the game will perform a few more logical checks to remove any completed lines in the pile and increase the score. As the score accumulates, the frame rate is increased, essentially giving the player less time to react to the game.

To draw the screen the game must convert the shape of a block to pixel data and move said pixels to the correct screen location. A similar process is performed to convert pile data to pixels.

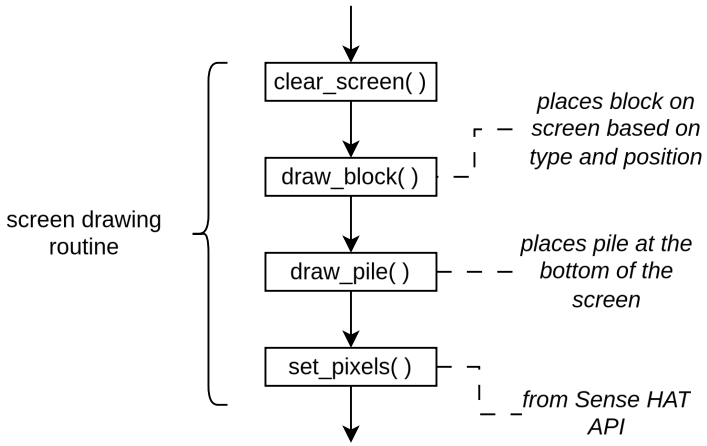

Figure 4.3: Steps for Frame Drawing During the Playing State

4.2 Accelerometer Input and Control Logic

The game uses the Sense HAT's accelerometer and joystick to provide an interactive and motion-based control system. The accelerometer detects the orientation of the device, allowing the player to move or rotate the game block by tilting the device. Also, the joystick is used to start, recalibrate and restart the game.

When the game begins or when the joystick's center button is pressed, the current accelerometer readings are stored as a baseline, representing the flat/natural position of the device. During gameplay, the program continuously compares the live accelerometer readings to this baseline to determine the direction of tilt.

To ensure accurate and stable control, thresholds are applied so that small movements are ignored. Only when the detected tilt exceeds a defined

threshold is an action triggered. Additionally, a stabilizing method is used to stop the game from switching directions too quickly. Once a tilt is detected, it does not immediately return to center when the device moves slightly back, but waits until tilt is clearly released.

The mapping of tilts to actions is as follows:

- **Tilt Left:** Move block left
- **Tilt Right:** Move block right
- **Tilt Forward (Up):** Rotate block left
- **Tilt Backward (Down):** Rotate block right

Chapter 5

Results and Discussion

5.1 Discussion

During the development of the Tetris game, the team encountered several technical and organizational challenges that influenced implementation and required careful management.

5.1.1 Technical Challenges

The game initially faced issues with tilt responsiveness due to incorrectly polled accelerometer input. This was improved by doubling the sensor polling rate and reducing the block falling speed, allowing smoother control.

To evaluate the effectiveness of these improvements, several tests were performed to measure the control responsiveness and accuracy. In the prototype implementation, players could perform approximately six actions per block, while in the final version this increased to fourteen actions per block, demonstrating a significant improvement in control responsiveness.

Additional experiments focused on measuring the accuracy of individual actions. As illustrated in Figure 5.1, both the *Move Left* and *Rotate Right* actions demonstrated improvements of approximately 10% compared to the prototype.

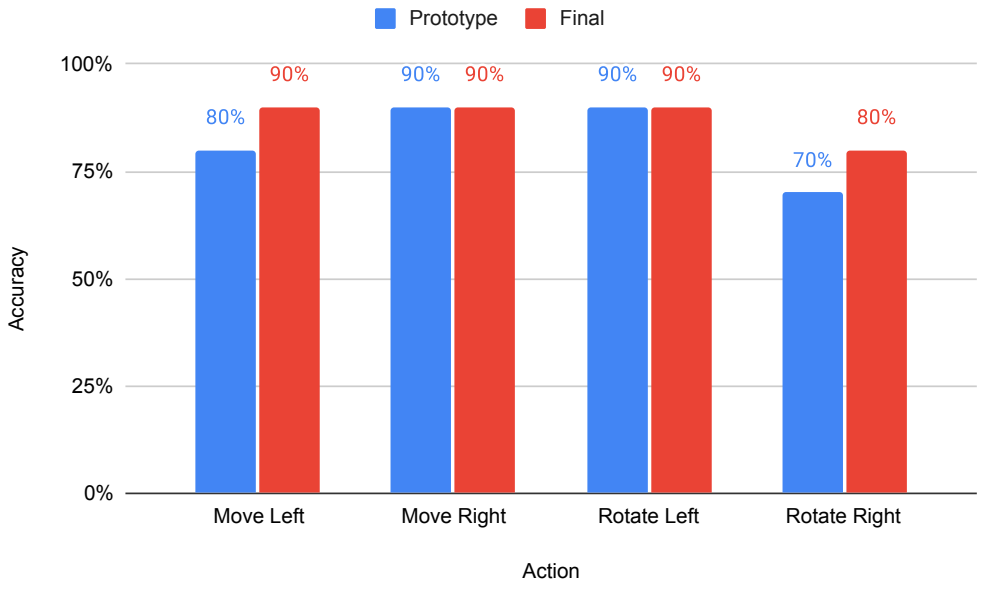

Figure 5.1: Comparison of tilting accuracy between the two different approaches.

Furthermore, the incorrect use of the magnetometer instead of the accelerometer was corrected during testing.

Other technical challenges included transforming sensor readings into meaningful game input, calibrating the accelerometer, integrating control logic with game logic, and dealing with hardware limitations such as the low-resolution 8×8 LED display.

5.1.2 Group Challenges

Coordinating tasks among multiple team members required dividing the workload into sub-teams based on skills and experience. Sharing the Sense HAT hardware posed logistical challenges, which were managed by implementing a schedule for effective usage.

Chapter 6

Ethical Considerations

Although *PiTris* is primarily a technical and recreational project, considering the ethical implications of its design and use is important. Even small educational games can raise questions about accessibility, fairness, environmental impact, intellectual property, and user health. Addressing these issues ensures the project is responsible, inclusive, and sustainable.

6.1 Algorithm Design and Ethical Lens

The algorithm for *PiTris* was designed with responsiveness, simplicity, and educational value in mind. Accelerometer-based tilt control was chosen to increase interactivity and highlight the integration between hardware and software. From a utilitarian perspective, the project aims to benefit a broad range of learners by making the game engaging and informative. However, we recognize that accessibility can be improved further, and alternative input methods may be necessary in future iterations

6.2 Accessibility and Inclusion

The tilt-based control excludes users with limited mobility. Future versions could include alternative inputs, such as buttons or voice commands, to broaden accessibility [4]. Testing with diverse users can help identify barriers and improve inclusivity.

6.3 Environmental Responsibility

Using physical devices generates electronic waste if not managed properly. Careful handling, reusing components, and responsible disposal can minimize environmental impact. Sharing hardware among multiple users and reducing the number of required devices further supports sustainability.

6.4 Copyright and Intellectual Property

Tetris is a copyrighted and trademarked game. This project avoids copying proprietary graphics, sounds, or branding, using only original or educational content. Respecting intellectual property ensures ethical use and sets a positive example for software development practices.

6.5 User Health and Screen Time

Simple games like this can become addictive due to their straightforward nature and the player's desire to achieve higher scores. To promote healthier play habits, future versions could include reminders to take breaks or completely remove the final score display. This would encourage players to focus on the enjoyment and challenge of the game rather than on continuous competition.

Chapter 7

Contributions

The project was completed through close collaboration among all team members. The following sections describe the individual contributions made during the development process.

7.1 Sotirios Aias Karioris

As team leader, Aias coordinated meetings, monitored deadlines, and worked with Julios on boot configuration and system architecture. He collaborated with Biruk and Arnab on the report and provided the initial program structure, which gave the team a clear foundation for the game.

7.2 Julios Fotiou

Julios led the development and worked with Aias during the initial setup. He created the accelerometer prototype that mapped tilt input to block movement on the LED grid. This prototype became the basis for the game's control system and guided later implementation.

7.3 Arnab Kumar Ghosh

Arnab monitored the team's progress and documented design choices and test outcomes. He worked with Biruk and Aias on writing and reviewing the report, helping ensure consistency and clarity in the documentation.

7.4 Biruk Aklilu

Biruk helped organize tasks, contributed ideas during discussions, and supported the development process. He worked with Arnab and Aias on structuring and drafting the report, particularly the introduction and purpose sections, ensuring clarity and coherence.

Although roles were assigned, all members remained involved across different areas. Teamwork, feedback, and mutual support were consistent throughout the project, contributing to its successful completion.

Chapter 8

Conclusion

This project presented the design and development of *PiTris*, a simplified Tetris game implemented on the Raspberry Pi with the Sense HAT. The finished prototype successfully linked accelerometer-based tilt input to block movement on the 8×8 LED matrix. By calibrating the accelerometer at the start, defining tilt thresholds through experimentation, and adjusting the sensor polling rate and block falling speed, we achieved accurate and responsive controls. Testing showed that each block could perform up to 14 moves or rotations with minimal input lag, confirming the effectiveness of the control scheme.

We encountered challenges such as sensor calibration, limited display resolution, and ensuring real-time responsiveness. These were addressed by establishing a baseline flat position and allowing joystick-triggered recalibration, optimizing the drawing order of blocks and the pile for better visual clarity, and synchronizing game logic with sensor polling. The result was smooth and predictable gameplay, validating our design choices.

Beyond the technical achievements, the project offered valuable hands-on experience in programming, hardware-software integration, and problem-solving. We also considered ethical aspects, including accessibility, copyright and intellectual property, and environmental responsibility emphasizing the broader impact of even small-scale technical projects.

Future improvements could focus on refining tilt sensitivity, expanding game logic, improving visual clarity, and exploring alternative input methods to make the game more accessible. Overall, *PiTris* demonstrates how compact hardware projects can combine technical skill, creativity, and educational value, providing both an engaging interactive experience and a

practical learning opportunity.

Bibliography

[1] Raspberry Pi Foundation, "Sense hat documentation," 2023. Available online: https://www.raspberrypi.org/documentation/hardware/sense-hat/.

[2] M. Kölling, "Educational programming on the raspberry pi," *MDPI Electronics*, vol. 5, no. 3, 2016. available online :https://www.mdpi.com/2079-9292/5/3/33=.

[3] S. K. Saha, S. K. Ghosh, and S. K. Saha, "2d-game development using raspberry pi," *International Journal of Computer Applications*, vol. 141, no. 9, pp. 1–5, 2016. available online :https://www.researchgate.net/publication/305686151_2D-game_development_using_Raspberry_Pi.

[4] J. K. Seale, *E-learning and Disability in Higher Education: Accessibility Research and Practice*. New York, NY: Routledge, 2nd ed., 2014. Available online :https://www.routledge.com/products/9780415629416.