Shapeville Design Report of Group 46

Zhenxuan Zhao 231220839，Zexuan Dong 231221652，Shuo Feng 231221294，
Weize Hu 231221766，Dachuan Zhao 231220046

1. Task Allocation

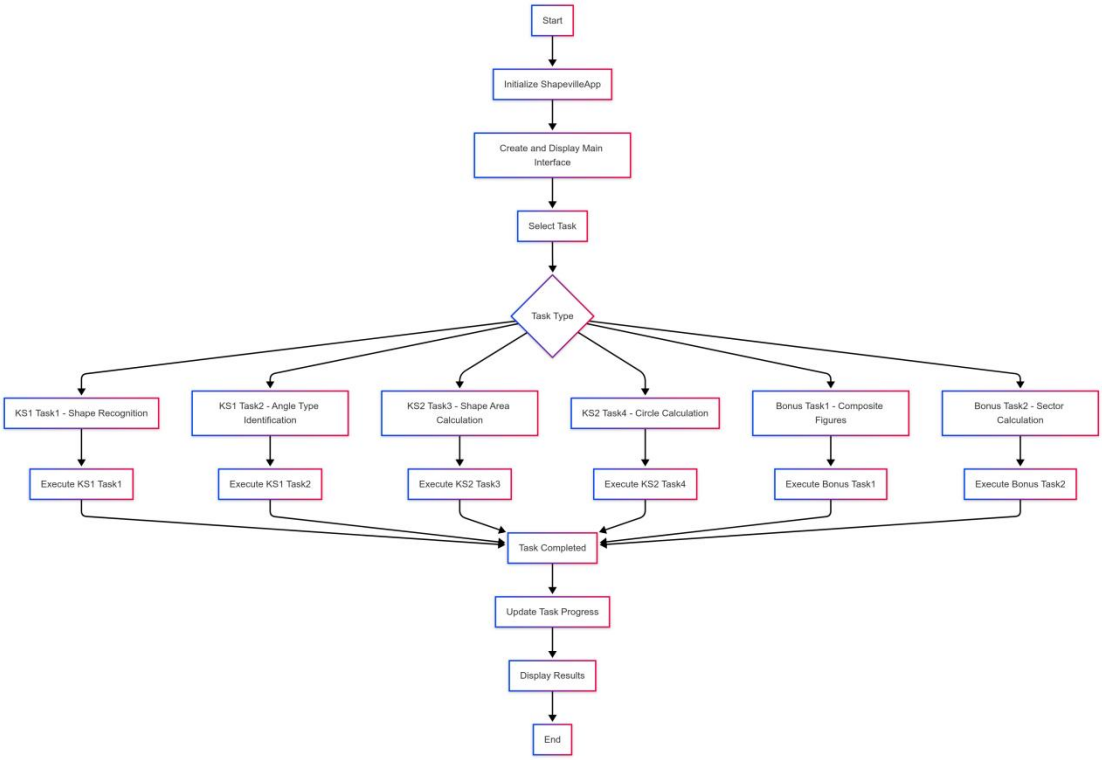
Group Member	Tasks
Dachuan Zhao	(1) GUI design and construction (2) preparation of flowcharts and class diagrams
Weize Hu	(1) Development of KS1 tasks (2) adding comments to the code
Shuo Feng	(1) Development of KS2 tasks (2) preparation of AI tool usage instructions
Zexuan Dong	(1) Development of Bonus tasks (2) compilation of user manuals
Zhenxuan Zhao	(1) Integration and testing of all code (2) preparation of task allocation and project reflection

All team members are involved in the design of classes and methods in the mini-project, which is organized by Zhenxuan Zhao.

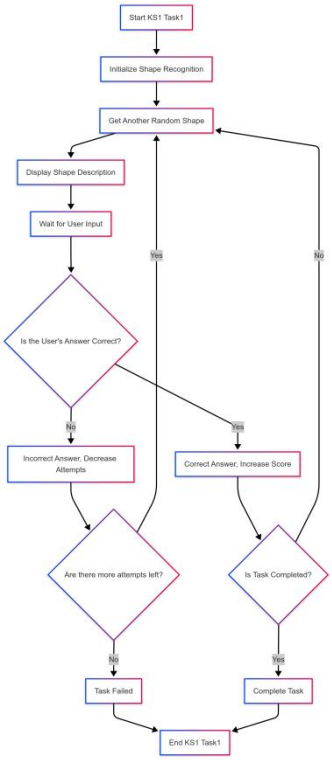
2.System Design

1.Flowcharts

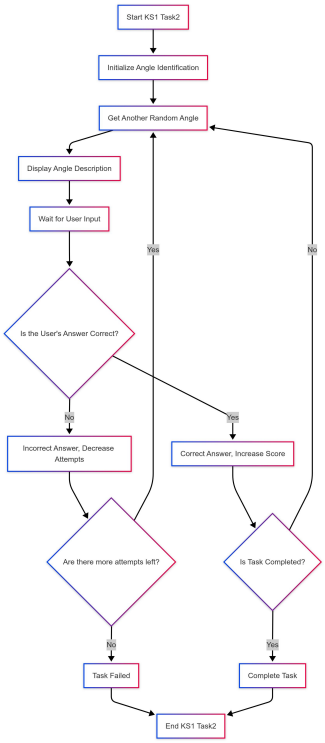
（1）Overall Process



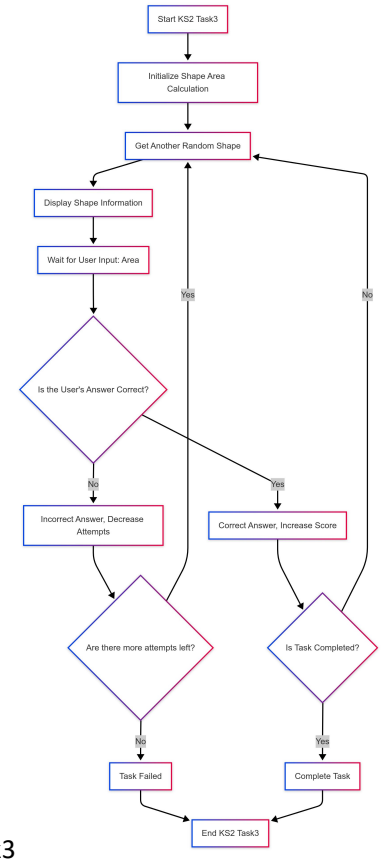
（2）Processes for each task



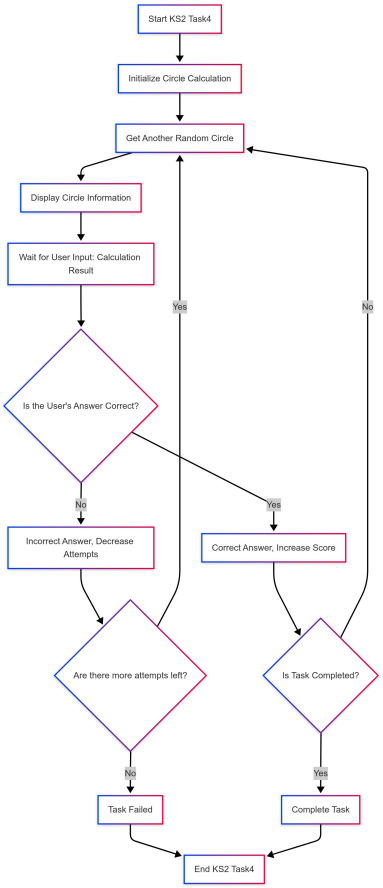
Task1



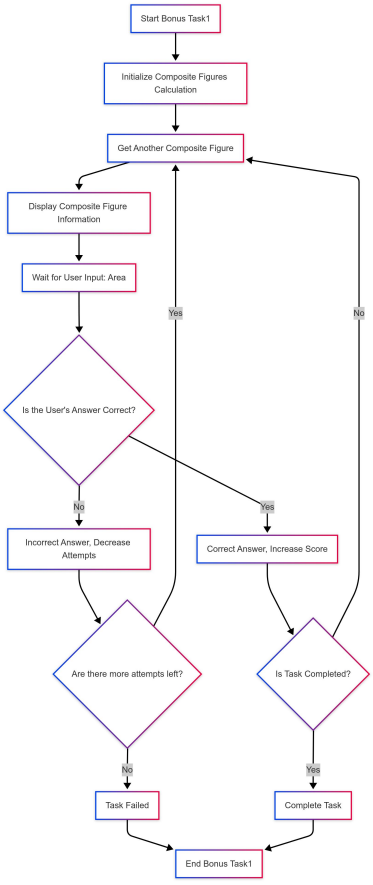
Task2



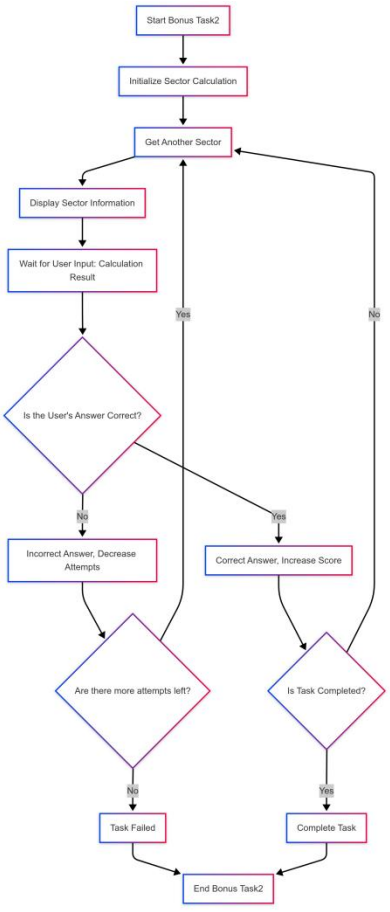
Task3



Task4



Bonus1



Bonus2

2. **Class Diagram**

3.Functional Design of Each Task

(1) KS1Task1: This module enables users to complete learning of graphic recognition, allowing them to choose either 2D or 3D graphics for study. The task is completed when the user finishes learning four graphics in either 2D or 3D. Each completed graphic will be scored according to the scoring system (basic scoring for 2D, advanced scoring for 3D), with feedback provided. Upon task completion, the progress bar will increase.

(2) KS1Task2: This module allows users to complete angle learning. Users can input any angle (must be a multiple of 10) in the angle input field and then match the input angle to one of the four angle types. Each successful angle selection will be scored according to the scoring system, with feedback provided. The task is only completed when all four angle types have been learned (learned angle types will be marked with a prompt), and the progress bar will increase upon completion.

(3) KS2Task3: This module enables users to complete learning the area calculation of four basic shapes. Users can start learning with any of the four shapes, and they have 3 minutes to calculate the area and input the answer. After completing one shape, they can return to the shape selection interface to choose another from the remaining shapes. Note that if three attempts fail, the formula and answer will be given, but the shape calculation is not marked as completed. A shape is only marked as completed if the calculation is correct (regardless of the number of attempts), and completed shapes cannot be reselected. Each successful area calculation will be scored according to the scoring system. The task is completed only when all four shapes are learned, and the progress bar will increase upon completion.

(4) KS2Task4: This module allows users to complete learning the area and arc length calculations of a circle. Users can start with either arc length or area calculation, and they

have 3 minutes to compute and input the result. If three attempts fail, the formula and answer will be given, but the circle calculation task is not marked as completed. A task is only marked as completed if the calculation is correct (regardless of attempts). Each successful circle area or arc length calculation will be scored according to the scoring system. The task is completed if either the area or arc length calculation is finished, and the progress bar will increase upon completion.

(5) Bonus1: This module enables users to complete learning the area calculation of six composite shapes. Users can start with any of the six shapes, and they have 5 minutes to calculate the area and input the answer. After completing one shape, they can return to the selection interface to choose another from the remaining shapes. If three attempts fail, the answer will be provided, but the shape calculation is not marked as completed. A shape is only marked as completed if the calculation is correct, and completed shapes cannot be reselected. Each successful calculation will be scored according to the advanced scoring system. The task is completed only when all six shapes are learned, and the progress bar will increase upon completion.

(6) Bonus2: This module allows users to complete learning the area calculation of eight circular sector regions. Users can start with any of the eight shapes, and they have 5 minutes to calculate the area and input the answer. After completing one shape, they can return to the selection interface to choose another from the remaining shapes. If three attempts fail, the answer and formula will be given, but the shape calculation is not marked as completed. A shape is only marked as completed if the calculation is correct, and completed shapes cannot be reselected. Each successful calculation will be scored according to the advanced scoring system. The task is completed only when all eight shapes are learned, and the progress bar will increase upon completion.

3.Rationality Explanation of Design Choices

1. Modularity and Maintainability

Layered Architecture: Adopts CardLayout to manage different task panels, switching interfaces via mainPanel to separate views from logic. Each task (e.g., shape recognition, angle calculation) is encapsulated as independent panels or inner classes (e.g., ShapeTask, AngleTask), enhancing code reusability and maintainability.

Inner Classes and Responsibility Division: Functional modularization is achieved through inner classes (e.g., ShapeList, ShapeAreaCalculationPanel). For example, ShapeList handles shape data initialization and management, while ShapeTask manages task workflows. This design adheres to the Single Responsibility Principle, reducing code coupling.

2. User Experience Design

Unified Visual Style: Predefined cartoon-style color constants (CARTOON_BACKGROUND, CARTOON_BUTTON) and fonts (CARTOON_TITLE_FONT) ensure consistent interface aesthetics and approachability. Hover effects (CARTOON_BUTTON_HOVER) and rounded borders on buttons enhance interactive feedback.

Intuitive Task Navigation: The main interface uses GridLayout to display task buttons, allowing one-click switching between learning modules (KS1, KS2, bonus tasks). Progress bars (JProgressBar) and real-time scores (scoreLabel) visually reflect learning progress to motivate continuous engagement.

3. Function Implementation and Algorithm Optimization

Graphic Rendering and Dynamic Feedback: Geometric graphics (e.g., protractors, composite shapes) are dynamically drawn via Graphics2D in paintComponent, combined with image resources (ImageIO.read) for visual enhancement. For instance, AngleTask uses trigonometric functions to render user-input angles in real time, illustrating geometric concepts intuitively.

Dynamic Question Generation: The Random class generates random parameters (shape dimensions, angle values) to ensure task freshness. For example, ShapeAreaCalculationPanel randomly generates dimensions for diverse calculation scenarios.

4. Error Handling and User Guidance

Input Validation and Immediate Feedback: NumberFormatException catches non-numeric inputs when answers are submitted, with JOptionPane prompting errors. For example, angle inputs only accept values between 0–360 in multiples of 10 to ensure data validity.

Multi-level Attempt Mechanism: Each task sets a maximum attempt limit (e.g., MAX_ATTEMPTS = 3), reducing remaining attempts on errors and displaying correct answers finally. This balances learning difficulty and fault tolerance to prevent user dropout.

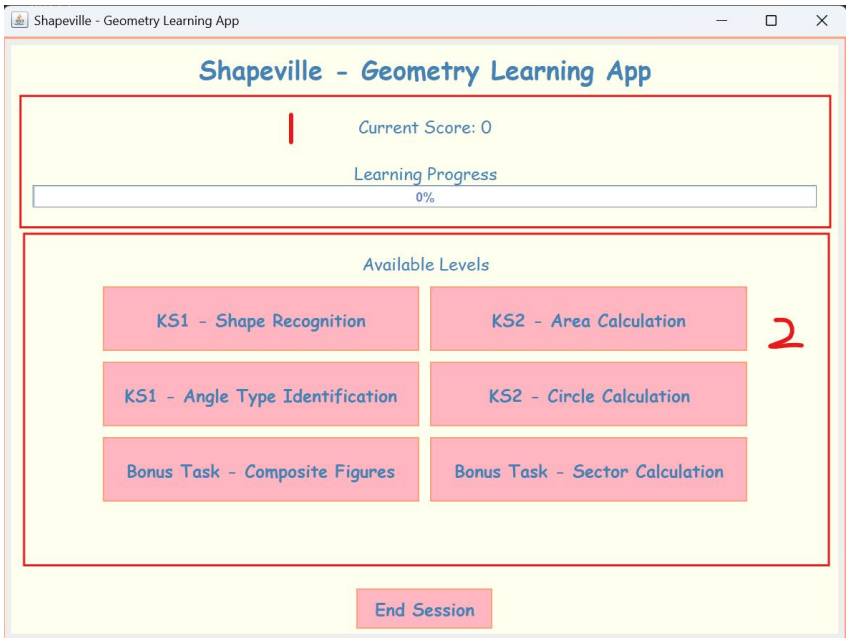
5. Extensibility and Flexibility

Flexible Task Configuration: A completedTasks collection tracks finished tasks for dynamic progress (progress) updates. Adding new tasks only requires extending mainPanel without modifying core logic.

Data-driven Design: Shape attributes (names, descriptions, dimensions) are encapsulated in the Shape class, allowing future extensions for new shapes by modifying ShapeList initialization logic.

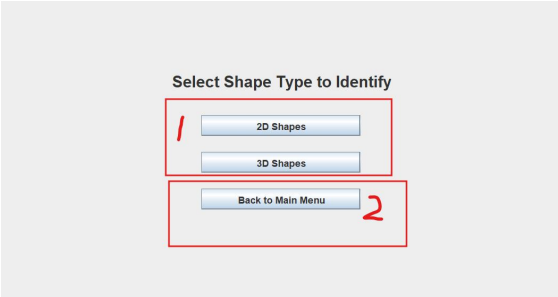
4. User Manual

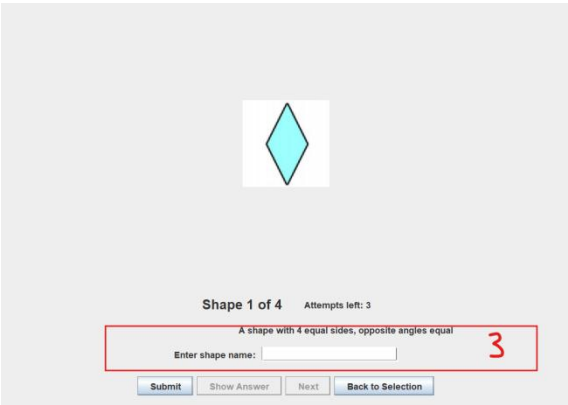
Step1) Enter the Main Window



Area 1 shows your learning progress and your scores.
Area 2 shows different tasks that you can choose to complete depending on your grade level, which include graph recognition, angle recognition, plane geometry area Calculation

Step2) Enter the task

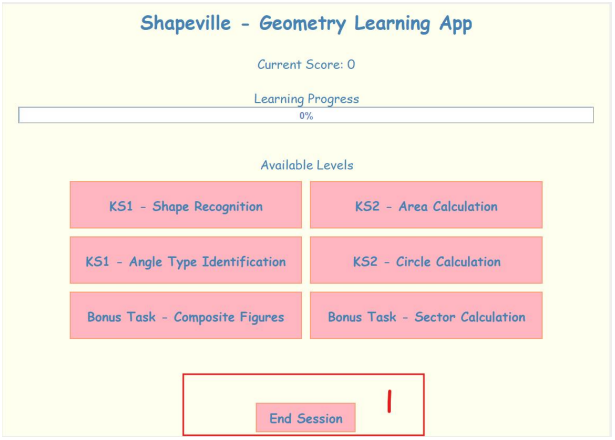




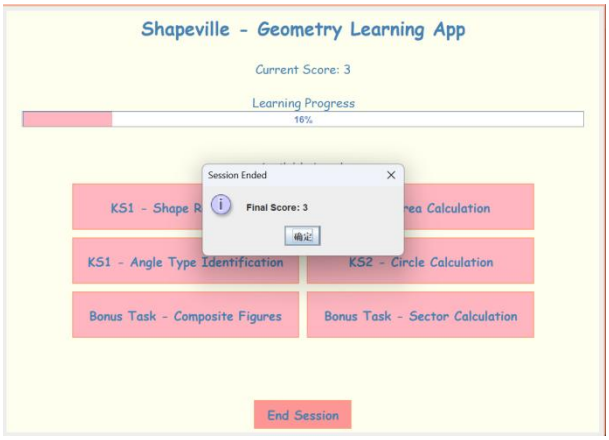
Once you enter the quest screen, you can choose task which you want to do in Area 1 and If

you want to go back to the main screen, you can click “Back to Main Menu” in Area 2. You can enter your answers in Area 3. You will have 3 chances to submit each question, and you will get points for correct answers, but the more wrong answers you get, the lower your score will be.
The interface for other tasks is as follows:

Step3) Close the shapeville



You can click “end session” to close the shapeville and it will tell you final scores.



5.Instructions for Using Artificial Intelligence and Other Tools

During the development of Shapeville, our team utilized various artificial intelligence (AI) tools and other assistive technologies to enhance development efficiency, optimize code quality, and improve user experience. Below is a detailed explanation of the tools used at different project stages and their specific application scenarios.

I. Usage of Artificial Intelligence Tools

1. Code Generation and Debugging

Tool Name: GitHub Copilot

Application Scenarios:

GUI Interface Design: When developing the 2D/3D shape recognition module, GitHub Copilot helped us quickly generate basic code frameworks, such as creating graphical interfaces using JFrame and JPanel, and implementing dynamic interactions through Swing components.

Algorithm Implementation: When calculating the area and circumference of circles, GitHub Copilot provided formula-based code templates (e.g., Math.PI * radius * radius) and assisted in handling floating-point precision issues.

Specific Example:

Prompt:

"Generate Java code for a 2D shape recognition GUI with user input and feedback, ensuring colorblind-friendly design."

Response:

GitHub Copilot generated JFrame-based interface code and suggested using high-contrast colors (e.g., yellow and blue) to enhance accessibility. We modified the response code to add dynamic refresh functionality to adapt to the display needs of different shapes.

2. Interactive Learning Logic Optimization

Tool Name: ChatGPT

Application Scenarios:

Task Flow Design: When designing the "Compound Shape Area Calculation" module, ChatGPT helped us sort out the logic flow for decomposing composite shapes and provided pseudocode examples to ensure flexibility in user input (e.g., supporting manual input or random parameter generation).

Specific Example:

Prompt:

"Design a step-by-step algorithm for calculating the area of a compound shape (e.g., a rectangle and triangle combined), with user input for dimensions."

Response:

ChatGPT provided a step-by-step algorithm and suggested using if-else conditionals to handle different shape combinations. Based on this, we implemented dynamic parameter input functionality and added exception handling to prevent invalid user inputs.

3. User Interface Layout and Accessibility

Tool Name: AI Design Tools (Figma + Adobe XD)

Application Scenarios:
Prototype Design: Used Figma’s AI-assisted features to quickly generate interface sketches, such as progress bars, button layouts, and color contrast tests.
Accessibility Checks: Verified interface colors using Adobe XD’s "Color Contrast Checker" plugin to ensure compliance with WCAG 2.1 standards, ensuring usability for users with color vision deficiencies (e.g., 14 types of color blindness).
Specific Example:
Prompt:
"Design a child-friendly progress bar with high contrast colors for accessibility."
Response:
Figma generated a progress bar design and suggested using borders as visual aids. After exporting it to Java code, we further adjusted the animation effects to enhance interactivity.

II. Usage of Other Tools

1. Code Management and Collaboration

Git + GitHub: Used for version control and team collaboration to ensure traceability of code changes.
Javadoc: Automatically generated API documentation to standardize code comments.

2. Documentation Generation

LaTeX: Used to write PDF reports and ensure format consistency.
Draw.io: Used to draw system flowcharts and class diagrams to assist in design documentation.

III. Key Prompts and Responses

Prompt 1:
"Generate Java code for a 2D shape recognition module that displays one shape at a time, accepts user input, and provides feedback with a progress bar. Ensure colorblind-friendly design."
Response:
GitHub Copilot generated the ShapeRecognitionPanel class code, including shape selection, answer input, and progress tracking.
Prompt 2:
"Design a formula-driven calculation for circle area and circumference based on user-selected parameters (radius or diameter). Include time limits and error handling."
Response:
ChatGPT provided a timer implementation based on the Timer class and suggested using try-catch blocks to capture non-numeric inputs. We applied this logic.
Prompt 3:
"Create a JavaFX animation for angle type visualization (acute, right, obtuse, reflex) with user-entered degree values."
Response:
GitHub Copilot generated angle visualization class code using animations to demonstrate angle changes. We adjusted the animation speed and added text descriptions to accommodate children with low attention spans.

6.Project Reflection

1.go well

(1) Recognizing that most users are children, we adopted CardLayout to manage different panels, enabling intuitive interface switching so kids can easily navigate between tasks. Buttons are designed with clear and straightforward labels for effortless comprehension and operation.

(2) The homepage features cartoon-style design elements—predefined colors and fonts enhance approachability, while hover effects and rounded borders on buttons improve interactive feedback, captivating children's interest.

(3) Frequent team meetings and open communication allowed us to incorporate everyone's perspectives, ensuring no tasks stagnated as members freely expressed ideas.

(4) Each member’s dedication and understanding fostered smooth collaboration, with inspired ideas elevating the design. For example, when recognizing angles, we designed a protractor to more accurately visualize angles, deeming a simple angle illustration with degrees insufficient.

2. not so well

(1) Initially integrating the project, we overused inner classes instead of outer classes, leading to overly verbose code that hindered readability and maintainability.

(2) Task division and communication issues arose during development—failure to balance team members' perspectives resulted in design imperfections.

(3) Inadequate review of project requirements caused extensive rework during integration testing. For instance, separate scoring systems in individual tasks were only standardized during final integration.

3. Areas for Improvement

(1) The project relies primarily on text and simple graphics, lacking multimedia elements like animations and videos. Adding these would vividly illustrate geometric concepts, enhancing learning engagement and comprehension.

(2) While basic accessibility considerations were included in the interface design, they remain incomplete.

(3) Some tasks may be too difficult or easy for all children. For example, KS2 tasks might overwhelm those with weaker foundations, while KS1 tasks could be too simple for advanced learners.