Jialing Wu

XRP Robotics Curriculum Design (Supplementary Kit）

– An Activities handbook based on the 7 Smart Teaching Principles

1. **Students' prior knowledge can support or interfere with new learning, depending on whether it is accurate, appropriate, and effectively activated.**
    - Allowing students to identify robots in life and describe programming tasks encountered in daily life (such as mobile phone automation and sweeper behavior).
    - Guide students to share their views on "programming" to activate and clarify previous knowledge.
    - Introduce the "You Know I Don't Know" wall to encourage students from different backgrounds to share experiences and blind spots.
2. **How students organize information into meaningful structures affects their ability to retrieve, apply, and transfer knowledge across contexts.**
    - Design a visual flowchart of "from problem to solution" to teach students how to decompose problems, select commands, and assemble logical structures.
    - Create a common bug prompt card, such as "What to do if the robot doesn't move", "It may be because the turning angle is wrong", etc.
    - Encourage students to write pseudo-code/annotations next to the code and explain their ideas in natural language.
3. **Students' motivation influences the choices they make, the effort they put in, and how persistently they engage with learning tasks.**
    - Before the maze challenge, students are allowed to choose the path style and design their own maze challenge story background (such as "robot delivery" and "jailbreak escape").
    - Introduce the role of "maze creator", let some students try to design the maze, and others to solve it.
    - Set up an achievement system (such as "Creator Badge" and "Most Stable Navigation Award") to improve motivation.
4. **To achieve mastery, students must acquire essential component skills, practice integrating them, and understand when and how to apply them in real-world situations.**
    - Use a scaffolded transition: natural language → pseudocode → block-based → Python.
    - Create a coding progression ladder with tiered challenges: e.g., first "Drive Forward," then "S-curve," then full maze.
    - After each new skill, add a brief recap session to reinforce integration.
5. **Focused practice that targets specific goals, combined with timely and actionable feedback, leads to deeper understanding and improved performance.**
    - Provide a structured feedback form (e.g., "What's working / What needs fixing / Suggested next step").

   - Use peer review and debugging exchanges: students test each other’s code and give suggestions.
   - Include quick mini debug challenges to model how feedback can guide immediate learning.
6. **Students’ learning is influenced by their stage of cognitive, social, and emotional development and by the degree to which the learning environment supports or challenges them.**
   - Set up "non-technical communication tasks" in the first two lessons, such as role-playing robot action flow, to reduce technical anxiety.
   - Provide differentiated learning resources: basic students use block diagrams, and advanced students consult API.
   - Design hierarchical group tasks to let experienced people drive inexperienced people and create a mutual learning environment.

7. **Developing into self-directed learners requires students to reflect on their learning, assess their progress, and adapt strategies to improve their outcomes.**
   - Introduce the three reflection questions at the end of each lesson (what did you learn today, what problems you encountered, and what you will do next time).
   - Add "My Debugging Log" at the project stage to record failure cases and resolution strategies.
   - Encourage students to set and adjust individual goals, such as "I hope the robot can avoid three obstacles this time".

Implementation Considerations

The activities should be introduced progressively and matched to students’ readiness. Flexibility is key—activities must accommodate diverse backgrounds and learning needs. Teachers will need time, support, and resources to prepare and deliver these activities successfully, and class time should include space for reflection and feedback.

Implications on Stakeholders

This approach benefits all stakeholders. Students become more motivated, engaged, and self-aware learners. Teachers gain deeper insight into student thinking and can tailor instruction more effectively. Curriculum designers are guided by a research-based framework, and schools may see improved learning outcomes and a more inclusive classroom environment.

Other Reference

- **How Learning Works: Seven Research-Based Principles for Smart Teaching**
- **How can you incorporate active learning into your classroom?**
- **Designing In-Class Activities: Examples of Active Learning Activities**