Teaching Statement: **Learning Games**

Ivan Ruchkin

[https://www.seas.upenn.edu/~iruchkin](https://www.seas.upenn.edu/~iruchkin)

The process of student learning is in many ways similar to mastering a multiplayer video game. Successful college courses and games place its participants into a motivating, inclusive, and productive environment, and these participants perform cohesive and increasingly challenging tasks. The crux of my teaching approach is to create learning games where motivated and engaged students progressively build their knowledge and skills. To me, the ultimate goal of these games is to develop Computer Science and Engineering students into system thinkers – professionals who understand and build computing systems holistically; that is, in terms of their interaction with mathematical laws, physical laws, humans, and societies [1]. System thinkers consider many viewpoints, are open to failure, and are committed to continually improving their thinking. I hope that through learning games my students will bring system thinking to a broad range of contexts in academia, industry, and government, becoming not only technical experts but also clear communicators and effective collaborators.

My teaching philosophy of learning games goes beyond simple gamification of learning (like progress tracking, visual decoration, and continual feedback) towards designing the teaching process and incentives after games. Consequently, my role as an instructor is not the central figure in a learning game – just like the game-maker is not the focus of the gaming process. Instead, I aim to enhance student learning indirectly through careful course design. Based on my teaching education from Carnegie Mellon’s Future Faculty Program and my experience across 8 courses, I believe that students become better system thinkers via learning games with three characteristics: (i) proper alignment of the parts, (ii) frequent stress-free assessment, and (iii) metacognitive skill transfer.

Successful learning games consist of educational elements that are well-aligned with each other: explicit learning goals determine educational activities, the results of which are continuously assessed according to the goals. Such well-aligned teaching is more likely to achieve its learning goals [2], including the development of system thinking. I achieve alignment by making sure that every instructional and assessment activity in a course, be it a lecture, a recitation, or a midterm problem, contributes to a specific and cohesive set of learning goals. To help students build up to these goals, I use instructional scaffolding to guide them through a progression of skills towards the higher levels of the well-known Bloom’s taxonomy of cognitive learning [3]: analysis, evaluation, and creation. For example, in the Principles of Software Construction course, the students analyzed what makes a well-designed object-oriented program before they were tasked with designing one of their own.

Once a learning game has its goals set and activities planned, frequent assessments should steer the course participants towards these goals. In practice, students often dread assessment, and sometimes this apprehension leads to disengagement, cramming, or even academic dishonesty. In contrast, my goal is create an inclusive environment where students can fail safely and learn from it. Casual videogames solve this problem by giving players several attempts to complete a task without significant repercussions. I emulate this characteristic in my teaching with low-stakes assignments [4] (e.g., weekly reading questions or one-minute written tasks), which help students get comfortable with making mistakes and learning from them. To further reduce the discomfort of assessment, I routinely discuss with the students a broader context of their learning. For instance, in the Introduction to Systems Programming, I have shown the students statistical summaries of their grades to date, highlighting the progress they have made and taking the responsibility for the instructional shortcomings, such as insufficient explanations or poorly designed questions.

Games often put players in unexpected situations where their past skills prove useful in new ways. In learning games as well, students practice solving new problems using what they recently learned. For example, they may apply well-known computational techniques, such as search and optimization, to complex real-world problems, such as image recognition. This skill transfer is particularly important in the modern marketplace that offers
computing jobs in diverse application domains from autonomous robots to gene sequencing. I enhance skill transfer by giving students ample opportunity to practice metacognition and self-evaluation in group exercises and reflective writing. Such exercises help students observe their thinking and carry over the qualitative insights to new situations [5]. Moreover, such exercises bring students awareness of their personal viewpoints and problem-solving approaches, thus helping examine the perspectives of others and become better system thinkers. For instance, in the last recitation of the Architectures for Software Systems, I asked the students look at excerpts from their past assignments and categorize the deficiencies of their past reasoning. This exercise enhanced the students' ability to observe, critique, and direct their own problem-solving process. In my experience, metacognition made students not only more knowledgeable, but also better communicators and team players.

To summarize, I design my courses as learning games with well-aligned parts, frequent low-stakes assessments, and metacognitive exercises to enhance skill transfer to diverse novel contexts. I believe that these games help students successfully acquire foundational skills in Computer Science and Engineering and, in particular, become better system thinkers. My application of this philosophy to teaching at Carnegie Mellon’s School of Computer Science has been acknowledged by the Special Recognition for Sustained Service in Teaching. In the future, I aspire to make learning games even more effective by combining them with other evidence-based teaching methods, such as flexible instruction time that accounts for student diversity. To me, the greatest reward of teaching is to share the modern academic culture with the future generations. In addition to competence, this culture encourages intellectual openness, curiosity, playfulness, and awareness of one’s own limitations and biases.

**Teaching interests.** Currently, I am comfortable teaching undergraduate and graduate courses on these topics:

- Data Structures and Algorithms, C/C++/Java/Python Programming, Object-Oriented Design, Operating Systems, Distributed Systems
- Discrete Mathematics, Formal Languages and Automata, Logic in Computing, Formal Verification and Model Checking
- Embedded and Real-Time Systems, Cyber-Physical Systems, Internet of Things
- Probability and Statistics in Computing, Data Analysis, Probabilistic Graphical Models
- Research Methods, Writing, and Practicum in Computer Science and Engineering

With some preparation, I can be ready to teach Computer Networks, Artificial Intelligence, Machine Learning, and Programming Languages.

**References**