Statement of Teaching Philosophy

Ivan Ruchkin, February 2017

The process of high-quality student learning is in many ways similar to playing a multiplayer video game. Just like a college course, a game places its participants into a motivating, safe, and productive environment: both players and students draw their inspiration from completing increasingly challenging tasks. The crux of my teaching and mentoring is creating **learning games** where engaged students progressively build their knowledge and skills. Specifically, I am passionate about training Computer Science and Engineering students to be **system thinkers** – professionals who envision their systems holistically; i.e., in interaction with physical laws, humans, and societies. I encourage students to explore beyond algorithms and programming, which remain the sole focus of many software developers today. System thinkers consider many viewpoints, are open to failure, and continually improve their thinking. I hope that through learning games my students will transfer system thinking to a broad range of professional contexts in industry and government, thus becoming not only computing experts, but also clear communicators and effective collaborators.

My teaching philosophy of learning games goes beyond simple gamification of learning (like using progress mechanisms and visual decoration) 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 a game maker does not jump in every time someone makes a move in their game. Instead, I aim to influence student learning invisibly through the game's design. In my opinion, students become better systems thinkers via learning games that have three characteristics: (i) *proper alignment of its parts*, (ii) *stress-free student assessment*, and (iii) *practice of skill transfer*.

Successful learning games consist of educational elements that are *well-aligned* with each other: explicit learning goals determine teaching activities, results of which are continuously assessed according to those goals. Such well-aligned teaching is more likely to achieve any learning goals, including the development of system thinking. For instance, once tasked with upgrading a legacy system, a system thinker would first consider its old and new requirements. I achieve alignment by making sure that every instructional and assessment activity in a course, be it 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 Bloom's taxonomy: analysis, evaluation, and creation. For example, in the Principles of Software Construction course, students analyzed what makes a well-designed object-oriented software system before they created their own one.

Once a learning game has its goals set and activities planned, *stress-free assessment* should navigate students towards these goals. In practice, students often dread assessment, and sometimes this fear turns an otherwise great learning experience into a race of cramming, subsequent surface learning, and even academic dishonesty. In contrast, my goal is to create system thinkers who are open to failure. Casual games solve this problem by giving players several attempts to complete a task without significant repercussions. I emulate this characteristic in my teaching: *frequent low-stakes assignments* (e.g., one-minute written questions after lectures and homeworks with multiple attempts) help students become comfortable with making mistakes and learning from them. To reduce the discomfort of assessment further, I routinely give students a broader perspective about their learning. For instance, I showed students statistical grade data for assignments throughout the Architectures for Software Systems course, highlighting the progress they have made to date.

Games often put players in unexpected situations where past skills prove useful in new ways. In learning games as well, students often practice solving new problems using their existing knowledge. For example, they apply well-known computational techniques (such as search and optimization) to complex real-world problems like image recognition. This *skill transfer* is particularly important in the modern marketplace that offers computational jobs in a broad range of application domains – from autonomous robots to bioinformatics. I enable skill transfer by giving students ample opportunity to practice *metacognition* and *self-evaluation* in classroom exercises and reflective writing. Metacognitive exercises enhance transfer by letting students understand general principles of their thinking and carry over qualitative insights to new contexts. Moreover, such exercises bring students awareness of their personal viewpoints and problem-solving approaches, thus helping examine perspective of others and hence become better system thinkers. For instance, in the final recitation of the Architectures for Software Systems course I asked students look at provided excerpts from their past assignments, and categorize mistakes in their past thinking. This exercise enhanced 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 teach and mentor by creating well-aligned learning games with frequent low-stakes assessments and metacognitive exercises to enhance skill transfer to diverse contexts. I believe that these games help students successfully acquire foundational skills in Computer Science and Engineering, and in particular become better system thinkers. I aspire to make learning games even more effective by combining them with other evidence-based teaching methods, like flexible instruction time to account for student diversity. To me, the greatest reward of teaching is spreading the modern academic culture to future generations. In addition to professional aptitude, this culture encourages students to develop intellectual openness, playfulness, and awareness of the limitations and biases of their own thinking. In the future, I would like to leverage interdisciplinarity in learning games to help students develop a well-rounded understanding of computing and apply it to other disciplines with maximal economic and social impact.