Eat, Code, Score: Coding Classes Serpentine Style*

A Nifty Assignment

Evelyn Brannock, Xin Xu, Wei Jin, Hyesung Park, Tacksoo Im
Department of Information Technology
Georgia Gwinnett College
1000 University Center Lane, Lawrenceville, GA 30024
{ebrannoc, xxu, wjin, hpark7, tim}@ggc.edu

Abstract

Using gaming as a pedagogical approach to engage students when teaching programming has been a strategy used by coding educators. Processing is a simple programming environment that was created to make it easier to develop visually oriented applications with an emphasis on animation and to provide users with instant feedback through interaction[1]. Although initially developed as a domain-specific language built on top of Java for creative coding applications, Processing[2] has matured into a robust framework. Introducing Processing to support game development can be used for students to easily learn the core concepts of OOP, even when they are not familiar with Java.

The focus of the lesson is two-fold: familiarize the student with the Processing environment, which supports graphical capabilities readily, and to gently introduce to the concepts of a Class and an Object in Object-oriented Programming (OOP), using an uncomplicated "Serpentine-style" game. By integrating the action of a recognizable game, students can gain CS skills while encouraging their engagement and continued attainment of a degree.

1 Materials

Students must download and install **Processing** [2], a free and beginner-friendly programming environment designed to create visually interactive applications that involve animation and real-time feedback.

Students will receive two sets of starter code, organized in zip folders named **Workshop 1** and **Workshop 2**. Each folder includes:

- A slide deck that guides students through the development process for that stage of the game.
- Starter code files to support hands-on learning.

Workshop 1 introduces Processing with a concise overview and features a demonstration of the Serpentine game, which students will aim to recreate.

Workshop 2 builds on this foundation, offering a step-by-step tutorial to implement the game using multiple Java classes.

No additional materials are required, except to download and install the open-source Processing Development Environment (PDE). Figure 1 is the sample game play. Full assignment details and starter code will be provided at the conference.

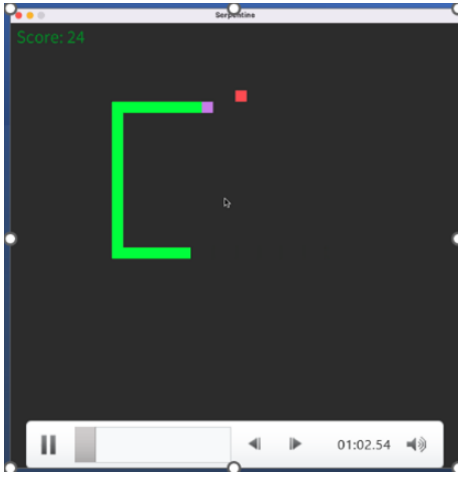

Figure 1: UI of Snake Game

2 The Assignment

Students are provided with two sequentially organized starter code packages distributed as compressed directories labeled **Workshop 1** and **Workshop 2**. Each package is accompanied by a corresponding slide deck that outlines the procedural steps required to advance the development of a game prototype.

Workshop 1 introduces the PDE, offering a concise technical overview and a live demonstration of the Serpentine game, which students are expected to replicate. The Serpentine game is a visually interactive application that focuses on animation. The snake (which is composed of segments) grows as it eats food (apples) and decreases as it starves. A score allows the student to achieve real-time user feedback.

Workshop 2 builds on the foundational concepts introduced earlier. This slide deck presents a structured, step-by-step guide that enables students to implement object-oriented design principles, including the construction of multiple interrelated classes essential for full-game functionality. The student will create initial implementations of the Apple, and Serpent classes. The will modify the supplied Serpentine class, emphasizing creation of a serpent, apples and movement.

3 Metadata

Summary	Students build two required classes and enable them in the overall program flow • A demonstration of the final resulting Serpentine game is provided by the instructor, who has a fully functioning version of the game as well as a video of the functioning game • Starter directions and the initial "game board" window are provided in Workshop 1 • Workshop 2 contains directions to create necessary classes to implement the snake and its food source. It also contains directions to modify the provided navigational flow program from Workshop 1
Topics	Classes and Objects, thinking in OOP, events, windowing
Audience	Mid CS1, Beginning Game Development

Difficulty	Low: The capability to recognize attributes and required but nominal. Limited knowledge of decisions is also a bonus.
Strengths	• **Introduces the value of reusable code:** Encourages learning further about the principles of well-designed , object-oriented code. • **Deep algorithmic and coding knowledge are not required:** Students are not required to understand advanced concepts or algorithms; this knowledge is encapsulated in Processing. • **Allows student to recognize that OOP principles are transferable:** Students can apply what they have learned to many languages. • **Inspires experimentation:** Students are encouraged to be creative and to discover additional features to implement in the game, from things as simple as changing colors, to more advanced, such as scoring mechanisms.
Weaknesses	IDE/API Overload: Processing uses its own IDE and API. For CS1 students, it could be slightly intimidating.
Dependency	Processing availability and documentation
Variant	Students can be challenged to add their own twists and customizations to the game

References

[1] Casey Reas and Ben Fry. Processing overview. `https://processing.org/tutorials/overview`.

[2] Processing Website. Processing development environment. `https://processing.org/`.