Using Incremental Worked Examples for Teaching Python and Game Programming

Teaching Python with Tkinter and Pygame Modules

Cheer-Sun D. Yang
Computer Science Department
West Chester University
West Chester, PA 19383, U.S.A.

Jonathan W. Yu
Information Systems & Statistics
Carnegie Mellon University
Pittsburgh, PA 15213, U.S.A.

Abstract—While world economy remains stagnant, enrollment in Computer Science suffers inevitably. A computer science educator must find innovative ways to inspire students for learning computing technologies. Active learning with “worked examples”, presented by Sweller and Cooper, has been proven to be an effective pedagogical strategy. However, when teaching Python with Game Building using the active learning approach, an instructor may run into several challenges. First, developing games requires certain familiarity with the programming language and mathematic. Without the prerequisites, students may not be able to participate in discussions actively. Second, learning game building can become extraneous cognitive load instead of incentives for learning python programming especially for students without knowing python. In this paper, an incremental model is presented for teaching python programming with game building. These incremental lab projects are easy to learn, easy to implement, and relevant with the targeted subjects. As a result of using the incremental projects, students become motivated; many, if not all, actively participate in the discussion about various subjects. The experiences learned are discussed. Student projects are briefly described. Finally, future work is included.

Keywords—python programming; active-learning; incremental lab experiments; worked examples.

I. INTRODUCTION

When world economy remains stagnant, enrollment in Computer Science suffers inevitably. A computer science educator must find innovative ways to inspire students for learning computing technologies. While the “worked examples” approach has been proven to be an effective tool for student-centered active learning, several challenges arise when applying “student-centered learning” model for teaching Python with Game Building. First, students may not have required perquisite knowledge to participate in discussions. Second, the incentives of building games for learning Python may actually become extraneous cognitive load especially for students without any python programming experiences. Without proper pedagogical treatment, the approach of integrating game building with python programming can hinder student-centered active learning.

During the past, using incremental lab experiments to achieve student-centered active learning has been demonstrated to be an effective approach for inspiring students [1][2]. Using the incremental approach, an instructor presents an initial problem for students to solve. A worked example will then be used to illustrate how to solve the problem. After this goal is reached, more requirements are incrementally added to the original problem. Students are expected to become more interested in learning more advanced skills. This is indeed another form of achieving “worked-example effect” [3]. As Sweller and Cooper defined in their paper, a worked example is defined as a step-by-step demonstration of how to perform a task or how to solve a problem [4][5].

This paper describes the approach of applying the incremental model for teaching python programming with gaming building using Tk interface (Tkinter) and Pygame. First, the integrated development environment (IDE), known as IDEL, is introduced. Then, the “incremental model” is presented in three areas, respectively: python, Tkinter, and 2D game building. Finally, the experiences learned and future works are described.

II. THE IDEL IDE

Throughout the course of teaching python and game programming, the IDEL IDE will be used. The IDEL IDE comes with the installation of python which can be downloaded from the official python web site at http://www.python.org/download/. While downloading python, an instructor can briefly introduce the online tutorial and library documentation available at the web site. Since there are two non-compatible versions of python, e.g., version 2 and version 3, one needs to be sensitive about the differences on syntax rules.

The IDEL IDE supports both the interactive mode and the scripting mode. Using the interactive mode, the IDE acts like a “calculator”. For example, one can enter “1 + 1” and then hit “Enter”; the IDE will display the resulting sum of 2 as illustrated in Fig.1.

The IDEL IDE also supports a scripting mode for which an editing window is displayed after clicking “File” tab > “New Window”. A python program can be entered in full and then executed by hitting Function key 5 (F5) on Windows PC. For changing the search path for a library module, the environment
variable PYTHONPATH may need to be changed. To check the search path, one can hit “File” > “Path Browser” to check for the current value of the search path.

In this course, focuses are on language features required for developing 2D games. When all students did not have the experience of using python programming language, most of them had experiences of using Java programming language. Therefore, the lecture can be customized to cover python programming language features that are different from Java.

As Python is a script language supporting dynamic data typing, for students who are familiar with Java, learning python is a fun ride. If an instructor can focus on the ease of learning and use, students will be inspired. In the following, the concept of “incremental examples” is illustrated with the list of topics to be introduced in the sequence of being introduced:

A. Basic Data Types, Operators, and Expressions

- Basic Data Types and operators: Python supports generic data types, including integer, floating point, complex number, and strings. Python supports basic arithmetic operators such as addition (+), subtraction (-), multiplication (*), and division (/) as well as floor (//), modulo (%), logic operators such as “and”, “or”.

- Following examples are used to illustrate the generic types and operators: (a) 10/5, (b) 5/10, (c) 5 / / 10, (d) 5.0 / 10, (e) 10%4 + 8/4, (f) 3 ** 10/3, and (g) “Test” * 3.

- Expressions: Python supports condition expressions that are similar to “3 < x < 5”. Following are exercises students work on using IDLE given a value for x, say 3: (a) 5 > x > 2, (b) 3 < x < 5, (c) 2 < x > 4, and (d) x < 5 > 2.

- Dynamic typing: Student can easily try the following examples using IDLE:

  ```python
  >> a = 3
  >> a = ‘test’
  >> a
  The resulting value for the variable a is “test” since python supports dynamic typing. The type of a variable is assigned during execution.
  ```

B. Control Structures:

- I/O statements: Before GUI library modules, e.g., Tkinter or Pygame, is introduced, students need to learn how to write input statement, i.e., raw_input() (version 2) or input() (version 3), and output statement, i.e., print (version 2), or print() (version 3).

- If statement: The minor syntactic difference of the If-statement between python and Java of C++ will not impose any psychological hurdle.

- For statement and range(): Python does not support the syntax of for-loop as in Java or C++. However, with the addition of using range() function, it is very user-friendly.

C. Sequence Types – List, Tuple, and Dictionary

- Examples about the sequence types are drawn from the online resource Wikipedia book [6]. The notation, comprehensions, slicing, and methods are fascinating to Java programmers. Students with Java or C++ programming experiences can pick up the ideas easily.

D. More Control Structures – functions, OOP

- Examples about user defined functions and object-oriented programming are covered. The incremental approach is used to illustrate the concepts with some worked examples [8].

E. Simple Non-GUI-Based Games

- The examples are used to illustrate how python can be used to develop games [8] including Tic-Tac-Toe and Reversi. The logic and the syntactic structure are explained. The book examples are developed without generic python language features without using any GUI module. Students will be encouraged to convert these non-GUI-based games to GUI-based games later.
Recursion, and (3) Photo Image File Handling. The ultimate goal is to cover skills required for developing game projects. The incremental projects used, respectively, are described below.

A. GUI Programming

- Basic Shapes & Drawing a box inside a window – As illustrated in Fig. 2, the drawing of a basic shape (oval, rectangle, polygon, etc.) is used to introduce the fundamental framework for GUI programming.

- Drawing Text widgets – The next step is to add some text widgets such as “Score” or “Display” to the previous example.

- Timing Control” – After the students become familiar with the basic GUI interfaces, the next step is to add animation to the example, e.g., moving the box without human intervention.

- Event-Driven Paradigm – A variety of event handling examples is used for this stage including: adding event handling to change the speed of the moving box, the moving direction of the object, or the color of the object.

B. Recursion

An incremental approach is used starting from python programming for generating a Fibonacci sequence, counting the number of vowels within a given text string, Sierspinski Triangle, and other fractal examples.

- Fibonacci sequence – The concept of recursion is illustrated using Fibonacci sequence as an initial worked example.

- Count vowels – For a given text string, one can write an iterative function as well as a recursive function. Students will be asked to think “recursively”. This is an exercise for students to complete.

- Sierpinski triangle – This is another worked example to illustrate recursion with Tkinter GUI programming. It provides not only the elegant taste of recursion, but also the pleasant effect of visualization. Ideally, the practice of active learning approach with incremental worked examples culminates the interests of learning more about python and game programming.

- Other fractal examples – Students will be asked to complete yet another fractal example in Figure 5 when the level of recursion is set to 2.

C. Photo Image File Handling

Game programming for 2-dimensional games may need to use image files as the background for a button widget, or the canvas widget. A worked example, as illustrated in Fig. 6, is certainly adequate. After a worked example is presented,
students learn how to use the PhotoImage widget in their projects.

![PhotoImage Widgets: Button & Background](image1.png)

**Figure 6 PhotoImage Widgets: Button & Background**

V. **PYTHON GAME PROGRAMMING EXAMPLES**

The incremental approach is used again for teaching python game programming. Students will follow the direction guide to complete a game with skeleton programs given to them at different stage. During the course of teaching python game programming, a simplified bubble shooting game is introduced. It is a simplified game similar to the one posted at the web page http://www.bubblesshooter.net/game1.php where the shooting is achieved with a bubble as the “bullet”. In the simplified version, a bullet is used.

- **Phase 1** - One row of target bubbles with the same color is used for the game as illustrated in Fig. 7.

![Bubble Shooter (Phase 1)](image2.png)

**Figure 7 Bubble Shooter (Phase 1)**

- **Phase 2** - Students will be asked to change the colors of the target bubbles and the target will only be removed when the bullet has the same color as the target.

- **Phase 3** - The target bubbles will begin to drop after a while and a new row of bubbles will be added at the top of the canvas.

![Bubble Shooter (Phase 2)](image3.png)

**Figure 8 Bubble Shooter (Phase 2)**

- **Phase 4** - Students can take this worked example and extend the program to support more functionality such as bouncing the bullet off the wall, removing all connected bubbles with the same color, or replace the bullet with a bubble.

![Bubble Shooter (Phase 3)](image4.png)

**Figure 9 Bubble Shooter (Phase 3)**

- **Phase 5** - An initial phase of a bubble spinner game is used to illustrate the strategies including rotating objects, bouncing the bullet off a wall. In this game, a cluster of bubbles is initially displayed to form a shape of a hexagon. More bubbles will be generated as the game proceeds. It is a simplified version of the bubble spinner game posted on the web page at http://www.deadwhale.com/play.php?game=774.

- A worked example of a bubble spinner (Fig. 10) is used to demonstrate some of the more advanced ideas such as generating hexagon bubbles, spinning objects, and bouncing a bubble.

![Bubble Spinner](image5.png)

**Figure 10 Bubble Spinner**
Each student is required to complete an individual project. Due to the time constraint of teaching this course in a 5-week summer session, students will be allowed to add features to an unfinished lecture example. But, grading will be based on the “actual” student work only; students need to state clearly which original program the student project is based on. In the following, several examples of student projects are presented. Students take about two weeks to complete the projects.

A. Super Duck Hunt:
Several ducks will fly out of left or right wall randomly. A player will move the mouse to “shoot” down a duck. For each player, there are five lives. For each duck missing, one life is lost. The game is over if the player loses five ducks.

B. Ping Pong Game:
A player presses “space bar” to start a game; “1” to move the paddles up; and “4” to move the paddles down.

C. Snake
A snake is shown as a series of connected squares. A type of food is randomly generated. A player will use the “Up”, “Down”, “Left”, and “Right” keys to change the moving direction of a snake. The goal is to “eat” the food. If the snake crosses its own body, hits the wall, or hits any barrier, the game is over.

D. Spinning Bubble Shooter
A cluster of bubble chains are given initially. When a bullet hits any bubble, the cluster of bubbles spins clockwise if the bullet hits the left side of the group; counterclockwise if the bullet hits the right side of the group. Once a while, an additional bubble will be added to the outer-most locations of each chain of bubbles.

E. Tetris
Tetris (Figure 15) is a “classic” game that can be implemented by using python with Tkinter module.
VII. EXPERIENCE LEARNED

This course was taught in a 5-week summer session. The first week is used mainly for covering Python programming with some game examples. The second week and the third are used for covering graphic user interface Tkinter with more game examples. The last two weeks are used for covering advanced topics such as rotating objects using pixels for the x- and y-coordinates, respectively, drawing of hexagon bubbles, bouncing objects. In the following, some experiences learned are described briefly:

- Course preparation – It takes excessive amount of time to prepare incremental programming examples for teaching game programming. Although there are plenty Python-based games available, most of them use Pygame library module instead of Tkinter library module. Also, these game programs need to be "tailored" for the purpose of using them incrementally.

- Teaching material – For teaching some other courses, it may happen that only a few on-line tutorials are available. In the case of teaching Python and game programming, it is just the opposite. There are many versions of on-line text material, sometimes too many; an instructor needs to customize the teaching material judgmentally with one caveat. Some require Python version 2, while some others require Python version 3. To be safe, an instructor needs to prepare to use both versions for demonstration and students need to be aware of the syntactic differences.

- Student evaluation criteria – Evaluation is somewhat difficult comparing with teaching without using student projects. However, assigning student projects can culminate students’ interests and conclude the course “with a high note”. A list of criteria is used including correctness, required features (a controllable component, a non-controllable component, animation, game-over, scoring, and restarting), complexity, user-interface design, and programming style, e.g., without unnecessary “hard-coded” program segments.

- Follow-up course – The teaching experience for this course demonstrates the feasibility of teaching with game building. It is strongly recommend that a Computer Science curriculum should be carefully designed to be student-centered. Moreover, an advanced course based on Python and game building can be used to sustain the culmination of interests created by this course. For example, Computer Graphics, Smart-phone Game Programming, and Artificial Intelligence Algorithm for Game Programming are prospective examples.

VIII. CONCLUSIONS AND FUTURE WORKS

Using the incremental worked example approach, an instructor can teach Python and game programming effectively. Students demonstrate high interests while learning Python as well as Tkinter and game programming. Assigning student projects culminates students’ interests and confidence about learning programming with game building. Once overcome the initial design phase, students expose tremendous innate esthetic capabilities in a short period of time.

Despite the seemingly successful effectiveness on promoting active learning, it requires to evaluate the effectiveness systematically about teaching with incremental worked examples. Indeed, this is only another example of applying incremental worked examples for inspiring students. For example, in the future, statistical methods will be used to study the effectiveness of the approach with incremental program samples for a quantitative study of the pedagogical approach.

ACKNOWLEDGMENT

C.Y. would like to express his gratitude for students in his Python and Game Programming classes at West Chester University. The feedback and the experiences are highly invaluable.

REFERENCES


