Engaging Students with Game Programming in Python

Hong Wang

***Abstract*—Python is becoming a popular programming language in teaching freshman programming courses. The author designed a sequence of game programming labs using Pygame to further help engage students and to improve their programming skills. The class survey showed that the adoption of Pygame is successful.**

***Index Terms*— CS1, Python, Game, Pygame, Assessment.**

I. Introduction

Large fraction of first year CS students is underprepared for hands-on programming course. In our program, around 20% of the students attending the first programming course were absent from the classroom and ignored homework assignments. Their enrolment in the class indicates their interest in the subject, but their absence shows that the delivery of the course might not be engaging enough. Various techniques can be used to make the classroom more exciting. Recent researches show that game and graphic themed programming courses have been successful [1,2,3,4,5]. Game design not only generated excitement, but also relates the programming course work to real world applications.

Packages and frameworks have been developed to help students in designing multimedia interactive games. Some are Microsoft based [6], and many are Java based [7,8,9,10]. With java, students need to learn OO first programming style. The instructor has to be cautious not to overload students with game projects.

However, in several papers including the author's previous paper [11] showed that python has many advantages in teaching first year CS students. The easy syntax and feasible interactive mode made python ideal in both academic and industry applications. Combining Python with game programming is promising. Attempts have been made by Csgraphics using a module library to teach graphics from day one [12]. In this paper, the author introduces a python course with game programming.

Pygame is a module library that suports drawing graphics and designing games [13]. Sophisticated games can be found on Pygame Website. A text book is given to students as reference material [14]. Using Pygame tools in teaching faces a few challenges. First, the students need to have not only some programming background, but also knowledge of graphical user interface before they can program real games. The course has to integrate game design with program development so that students can achieve better programming skills. This paper describes an innovative approach in engaging students using Pygame from day one and analyzes the results to help future development.

II. Course Design

The course tries to combine the flexibility of python and the attraction of game design. This section introduces a sequence of labs and homework assignments that helps achieving this purpose.

A. Basic Programming Skills

Before starting to design games, students need to learn basic programming skills. The course needs to blend game design with early learning process. True interactive GUI game design couldn't happen before students can fully handle at lease sequential, branch and loop programming. These could cost half of the semester. This phase is a good opportunity to teach students how graphics are achieved in Pygame. The concepts include coordination system, pixels and refresh rate. It is straightforward to show student some simple drawing functions to draw circles, rectangular and lines. These graphics functions are then used in sequential and branch program practices and assignments.

B. Branch

Instead of printing strings, programming assignments were made more interesting by drawing shapes. If a certain condition is met, then a certain shape can be displayed. A more challenging homework is to draw a number of random shapes by using Python's random number generating library. After generating a random number, the program then draw a certain shape based on the number. As example, if the number is 1 draw a circle and if number is 3 draw a rectangle or a line. These simple exercise are visually appealing than simply displaying strings. Figure 1 shows the result of this assignment.

Manuscript received July 2, 2009.

Hong Wang is with the Computer Science and Engineering Technology Program, Engineering Technology Department, The University of Toledo, Toledo, Ohio 43606 USA (e-mail: hong.wang2@ utoledo.edu).

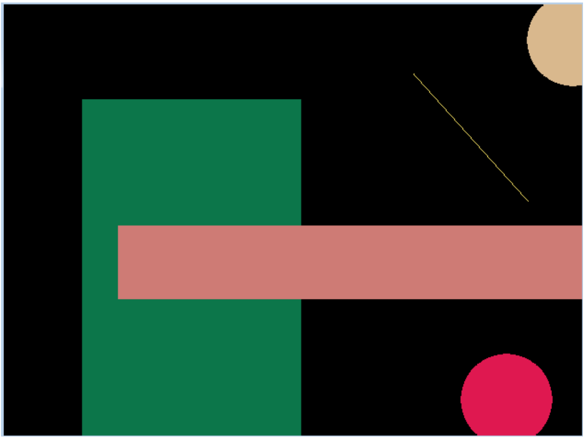


Figure 1. Branch using shapes

Figure 2. Loop and Animation

C. *Loop*

Since Pygame animation relies on refreshing the Pygame window using while loop, it is the best for students to start learning animation after finishing loop chapter. In a while loop, update() or flip() functions are used to keep on refreshing newly drawn graphs or pictures. Though it is possible for students to animate the shapes they practiced before, it is more interesting to work with images or photos.

In Pygame, pictures can be displayed with blit() function. The blit() function takes the pictures' coordinates as a tuple argument. Students can manipulate the coordinates of the objects to move them. Flip () function refreshes the screen to display the newly drawn picture.

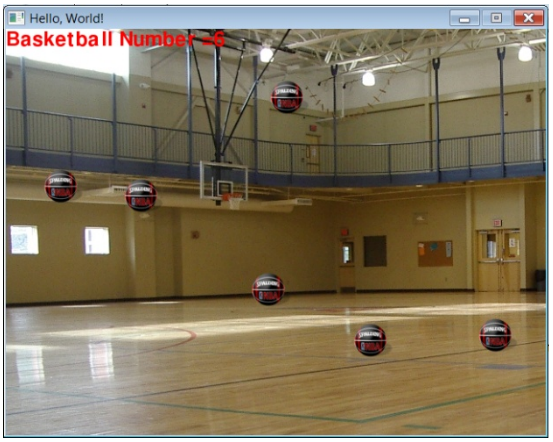
Events are detected by a for loop nested inside the while loop. A program can close a Pygame window, by detecting a close window event in the for loop.

Understanding the role of these loops is critical for students to understand GUI display, event detection and game design. Programming concepts are reinforced in the design process.

Assignment in this chapter is to design an animation. A background image should be displayed first, and then an object should move on the background. Carefully calculating the coordinates is the key to make objects move as desired. However, branch structures are needed if the object makes any movement beyond simple linear movement. Combining looping and branching structures to design the animation is both challenging and engaging. Figure 2 shows a student's homework showing a raptor flying over a town square.

D. *File I/O*

In previous semesters when we use C to teach freshman programming, homework for file I/O was to read a file in a certain format and print in some other format. With Pygame, we could use GUI to make the assignment more interesting. Students are given an input file that specifies the position and the name of the shape to draw. Their Pygame program should be able to read and draw the corresponding shapes at the specified coordinates. This exercise involves a few steps. First the program has to read the file content and then split it to a list (Python array) for further processing. Then the name of the shapes can be used in conditional structures to determine which shapes to draw. The coordinates and colors are passed as tuple arguments to the drawing functions. The author used random numbers and shapes last time the course was taught. It might be more interesting to draw something meaningful such as sky with sun and cloud or robot shape. However, it might take the instructor more time to demonstrate and to grade. Another option is to specify file names of graph files for students to display. The final display could tell a story using graphics. Either of those options could make file I/O more fun to do and easier to understand.



Basketba ll Number =6

Figure 3. Multiple bouncing basketballs as objects

E. *Software Objects*

The most important challenge of the class is to teach object orientation. Unlike Java where everything has to be an object, python supports both Procedural programming and object orientation. Traditional way in introducing objects to new programmers is to use names such as a "car" object. As example, one can define a car class and then generate car objects that have "gas" attribute and "run" method. Since everything is text based, it is still difficult for student to understand the point of OOP.

Using graphics, one can represent objects using something looks like an real world object. The instructor shows the students a non-OO version of a program that display a fish object by clicking the mouse in Pygame screen. This example can then be modified to an OO program. The two programs do exactly the same thing. When mouse is clicked inside the Pygame window, a fish is displayed. However, the OO program is easy to extend. The teacher then illustrates how to generate and display multiple fishes as objects from the same class with mouse click. Each fish has its own position attributes.

The homework exercise requires students to implement a similar program where the generated objects are not only displayed but also moving on their own courses. The game play made it fun to relate software objects with real world objects in OO programming. Figure 3 shows a student's work. Multiple basketballs bounce around randomly in a basketball field.

F. *Object Oriented Programming*

The last chapter to teach is inheritance and polymorphism. The fish Class in the last chapter can inherit from a more general fish Class. Two subclasses Coral fish and Shark inherit this general fish class. Coral fish and Shark classes override the move method in fish class. Coral fishes move randomly around a coral reef background while the shark is controlled by keyboard to try to catch the coral fishes. Once the shark meets a Coral fish, the Coral fish disappears from the coral reef (figure 4).

Homework assignment requires students to modify the program so that there are one or two sharks hunting around the coral reef. A player controls a Coral fish using keyboard. If the Coral fish could swim from one side to the other side, the player wins the game. Figure 5. shows a student's work.

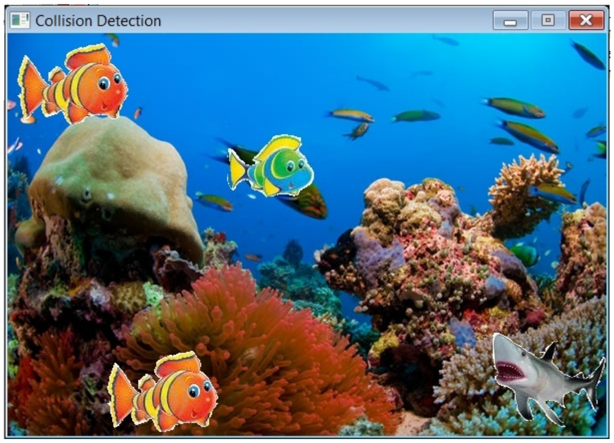


Figure 4. Coral fish

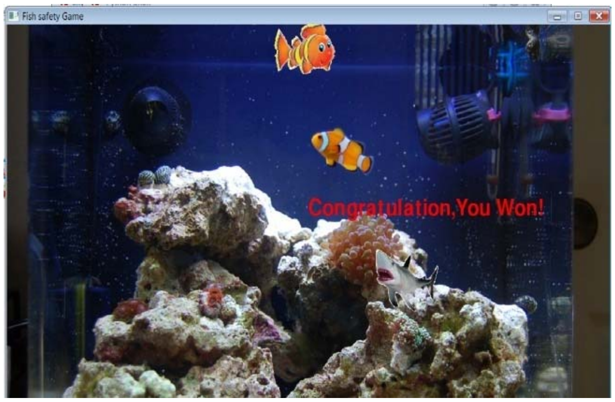


Figure 5. Coral fish hunting game

III. Class outcome and Feedback evaluation

From the teacher's point of view, the adoption of Pygame was a success. Students were able to program more sophisticated codes and achieved better grade than previous non-Pygame classes. The student's side of view is obtained using the following survey. In the following table, A is strongly agreed and E is strongly disagreed.

1. Do you think that doing pygame programming helps gaining an understanding of how to program?

A(50%) B(50%) C(0) D(0) E(0)

2. Do you think that doing pygame programming helps gaining an understanding of Object Oriented Programming?

A(33%) B(50%) C(17%) D(0) E(0)

3. Was working on a game more or less enjoyable for you than assignments you have done in other classes?

A(50%) B(50%) C(0) D(0) E(0)

4. Did the fact that it was a game lead to you putting in more time or effort than you would have for a different type of project? (A is much more time, E is much less)

A(33%) B(17%) C(33%) D(17%) E(0)

5. Did you feel that you had sufficient flexibility to create what you wanted to?

A(50%) B(33%) C(17%) D(0) E(0)

6. Did you find that understanding the pygame was a challenge?

A(33%) B(50%) C(17%) D(0) E(0)

7. By the end of the semester did you feel that you understood how things worked in pygame?

A(33%) B(50%) C(17%) D(0) E(0)

According to the survey data, students are all positive about the class. They not only had fun but also gained help from Pygame in understanding programming and OO concepts. Design of the class has made sure that the Pygame do not incur too much extra work and this is confirmed by the student's feedback. This is only the second time the class is offered with python and Pygame. The author expected that in the future, more "A" feedback will be given to the class as more experiences are accumulated.

IV. Discussion

Instead of using plug-ins to handle game design, students in this class have to build the programs from scratch. The

Pygame lessons are designed to help students in mastering the programming topics of each week. Pygame examples and exercises needs to be properly synchronized with the python programming course flow. Course outcome evaluation and student's feedback confirmed the author's approach. The success of this first programming class will help us design a better CS2 course where the focus will be problem solving and data structures.

REFERENCES

[1] E. Sweedyk, M. deLaet, M. C. Slattery, and J. Kuffner, "Computer games and CS education: why and how," in Proceedings of the 36th SIGCSE technical symposium on Computer science education , St. Louis, Missouri, USA , 2005, pp. 256-257.

[2] U. Wolz, T. Barnes, I. Parberry, and M. Wick, "Digital gaming as a vehicle for learning," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 394-395, 2006.

[3] K. Becker, "Teaching with games: the Minesweeper and Asteroids experience," Journal of Computing Sciences in Colleges , vol. 17, no. 2, pp. 23-33, Dec. 2001.

[4] J. D. Bayliss and S. Strout, "Games as a "flavor" of CS1," ACM SIGCSE Bulletin, vol. 38, no. 1, pp. 500-504, Mar. 2006.

[5] S. Leutenegger and J. Edgington, "A games first approach to teaching introductory programming," ACM SIGCSE Bulletin, vol. 39, no. 1, pp. 115-118, Mar. 2007.

[6] K. Sung, R. Rosenberg, M. Panitz, and R. Anderson, "Assessing game-themed programming assignments for CS1/2 courses," in Proceedings of the 3rd international conference on Game development in computer science education, Miami, Florida , 2008, pp. 51-55.

[7] A. Luxton-Reilly and P. Denny, "A simple framework for interactive games in CS1," in Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, 2009, pp. 216-220.

[8] D. Frost, "Ucigame, a java library for games," in Proceedings of the 39th SIGCSE technical symposium on Computer science education, Portland, OR, 2008, pp. 310-314.

[9] M. Kolling, Introduction to Java Programming Using Greenfoot, 1st ed. Upper Saddle River, NJ, USA: Prentice Hall Press , 2008.

[10] W. P. Dann, S. Cooper, and R. Pausch, Learning to Program with Alice, Brief Edition. Upper Saddle River, NJ,USA: Prentice-Hall, 2006.

[11] H. Wang, "From C to Python: Impact on Retention and Student Performance," in Proceedings of The 2009 International Conference on Frontiers in Education: Computer Science and Computer Engineering (FECS), Las vegas, USA, 2009, pp. 170-174.

[12] M. H. Goldwasser and D. Letscher, "A graphics package for the first day and beyond," in Proceedings of the 40th ACM technical symposium on Computer science education, Chattanooga, TN, 2009, pp. 206-210.

[13] Python Game Development. 2009. www.pygame.org. Last accessed on: July 31st, 2009.

[14] W. McGugan, Beginning Game Development with Python and Pygame: From Novice to Professional . Apress, 2009.