CS 152 Computational Thinking: Scientific Applications
Section A - Fall 2021
Final Project Requirements and Specifications

The Internship Interview: Developers Choice
Preview Check in: December 6th in class (collaborative lab)
Project Code and Report Due: Dec 10, 2021 11:59pm

This project is an assessment of your ability to meet 2 major course learning objectives:
LO 2. Students can convert a problem statement into a working solution that solves the problem.

LO 3. Students understand abstraction and can break down a program into appropriate procedural and object-oriented components.

Problem Statement:
You have been selected to interview for an exclusive internship with a top researcher in your field (e.g., psychology, biology, physics, economics, etc.). They have asked you to bring an example of your programming skills as applied to a dataset or simulation problem of your choice to demonstrate that you have the ability to work through a complex problem and develop a solution or model. They are particularly interested in your ability to demonstrate your grasp of abstraction and the integration of procedural and object-oriented programming into your solution or model. In addition, the researcher has specified they need to know you understand, can demonstrate, and communicate about key programming concepts in your solutions (see list below).

Your report documents should be as professional and seamless as possible for the researcher to be able to run and understand. This means your main code should ideally run from one file, using file management methods to create a smooth interaction experience for the researcher. You will be provided with a report template that is used in the researcher's lab.

The goals of this final project are to demonstrate:

- Understanding and mastery of course coding skills, CS concepts, and best practices in code organization and documentation.
- Creating complex, interactive program(s) that adapt modular elements from previous projects in novel and creative ways.
- Applying best practices in top-down OO design methods in the program composition.
- Understanding of the ways in which researchers and programmers may inadvertently introduce biases and errors into their solutions and ways in which this might be prevented or mitigated.

Instructions to Run the Program:

To run the program, go to the directory with all the game files in the terminal and run "game_main.py" with the command `py game_main.py`. You will first be asked to select a color for your car in the terminal. Write down any color and press Enter. The game will then open up in a Zelle Graphics window. Click on the window to make sure that the checkKey command for the controls work and then play the game.

Project Abstract:

This project is about a car game. The user controls the car to jump over obstacles and collect coins. The coins and the number of lives that the user has left are represented on either of the top corners of the screen and the user needs to collect as many coins as possible before running out of lives. In this project, I used inheritance to make the graphics objects since they have similar characteristics; this means that I created a parent 'Thing' class with some general attributes (characteristics of each object I made that are similar) and methods (functions of each object that are similar) and inherited them into the child classes (which are the shapes and the complex objects like a car or a tree that I made). I used modularity to achieve each function in the game and put them into a separate file called "game_functs.py", this file has a total of 6 functions performing different tasks within the game. This project also needs to model collisions between objects and so I used a previous collision file given to me during a previous project. I also made a collision file of my own called "new_coll.py" to model some specific interactions in the game like the one between the car and the coins. When the user runs the program, they'll be asked to choose a color for their car and then the game will begin.

Project Design Sketch/Image:
I included a video of my simulation as an attachment to this submission.

Important things to note:

The coin counter refreshes when the user makes it to the end of the screen

The lives counter is live

The user needs to click on the screen after choosing a color or else they won't be able to jump

The project must demonstrate the following skills and concepts:

- **Modularity** making functions work as independent units that you can re-use at any time. This means making small, modular functions with keyword parameters that contribute to larger composite functions with test functions for composite shapes and scenes.
- **User Interaction:** Use conditionals, user input prompts, and command-line arguments to add nuance and interactive features to your project.
- **Best practices for code organization:** File docstrings, function docstrings, file comments, instructions for users related to interactive features (command line or user input text), Effective use of main code to if __name__ == '__main__': in files.
- **Function complexity:** randomization, return values, and multiple assignments, use lists to effectively organize multiple complex objects, dictionaries, sets, tuples.
- **Plots, graphs, visualizations:** appropriate graphics to effectively communicate results of analysis with concise but meaningful interpretations and explanations for your audience.

Choose 2 out of 3 Project Features to integrate into your solution/model:

- **Complex Objects and Scenes:** One simulation with a Zelle complex object (e.g., collections of Zelle graphics primitives). Each complex object will have a function that initializes it (__init__) . Complex objects that change will have a function that controls it (e.g., changes it, animates it, etc.) and a function that tests it. (I created complex objects and scenes)

- **Inheritance Classes:** One part of the project that uses parent and child classes with accessor and mutator methods to separate how data is stored from how it is used in Classes and Objects. (I did use inheritance in my code)
- **Recursion:** At least one use of a recursive function to perform some type of analysis in your solution.

The project report and program files that you will submit for the interview will demonstrate that you have the following skills necessary for the internship position:

1. **Provide a brief description of the specific programming processes, concepts, and/or approaches to your potential employer so they can quickly understand and be persuaded of your programming skills based on the key components of your work.**

Object Oriented Design: I used OOD to make the graphics objects in my project. Using OOD helped me make the code for the objects more concise since I was reusing methods and attributes and also made it easier to initialize objects when I was playtesting my game.

Modularity: I made the game in separate pieces. First it was just a car on a road, then a car with coins that it could collect, then a car that collected coins that could jump, then a car that had to collect coins while jumping over objects, and then finally a car with lives and a coin counter that had to do all those things. Using modularity made it so that I knew where the problem was instead of continuously having to revise all my code.

User Interaction: The game involves user interaction through having the user pick a color and by having the user control the car as it navigates the obstacles.

Function Complexity: The game has a high level of function complexity. I used return values to reference objects in different functions, I used functions inside other functions and imported functions from a series of different files to make the final main function. An example of this would be the coin collision function that erases coins created by the coin generator function and then counts the erased coins in the coin counter function. All of these different functions are connected together in a complex way.

Plots, Graphs, Visualizations: The game is essentially a visualization of all the code. The car, the coins, the obstacles are all coded in and represented in the window that the user sees. Even the interactions the car has with the edges of the window, the floor, or the obstacles is all coded in and represented visually using collisions and other things that happen when certain conditions are met.

2. **Explain the scientific processes, datasets, and concepts that you have embedded into your process.**

Collisions & Physics: The objects interact with each other in an interesting way. A function constantly monitors the position of the objects and sees if they are close enough to be visually colliding. The function then performs an action; this could be resetting the car, making a coin disappear or making the car bounce. All of this has Physics at its core.

Even the movement of the car, its velocity and acceleration are mapped using Physics equations and then represented on the user's screen.

Organization: The organization of the code makes it so that it is easy to tell where the problem lies. If something weird happens when the car collides with the coin then the user can go check the car-coin collision function. Similarly for any other interaction, the code being broken into so many small parts helps to debug it and keep it working.

Creating Shapes: Creating shapes was probably the harder part. I first had to understand Zelle Graphics and then go through a great amount of trial and error to create the car, to make sure the tyres were in the right place, the lengths and radii of the objects were reasonable and then that there weren't any problems with performing the methods inherited from the parent class.

3. **Provide a critical evaluation of your design and what you might do differently in the next iteration. This is an analysis of your own solution and its rationale, methods, and implications.**

I would make a dictionary to have different positions for the obstacles and then corresponding positions for the coins to make the game more interesting.

I would use pygames to introduce more user controls like moving forward and backward and maybe doing a backflip.

I would like to finish the lag that happens in the game (times the car dips down beneath the floor or gets stuck while moving forward). I think this had a lot to do with the dt values in the code. I tried experimenting with them and that changed things a lot but I still couldn't fix it in time.

I feel like the design for the collisions and the corresponding complex reactions like the lives disappearing or the coins being counted was nice, but I could have done a better job by making it so that the coins would refresh live as well.

4. **Provide an example(s) of how you have written concise, precise, easy to read code and comments so that it can be easily maintained using good coding style and organization.**

```
def coin_counter(win, coin_counter, message = None):
    '''this function updates the users score at the top left of the screen after they finish every game level'''
    #seeing if there is already a score on the top left of the screen, if there is one, it undraws it
    if message != None:
        message.undraw()
    #writing the new score in the same position according to the player's latest score
    message = gr.Text(gr.Point(50, win.getHeight()-680), coin_counter)
    message.setFill('White')
    message.setSize(10)
    message.draw(win)

    return message #returning the message so that i can use it in the parameter of this same function
```

Important design decisions and the importance of any code is noted down in my comments. There are also doc strings for every function and the functions are spaced out evenly with two spaces in between.

```
#this is how i keep count of the coins collected, if they're collected, they're undrawn and their drawn status is == False
            for i in range(5):
                if coin_label[i].drawn==True:
                    continue
                else:
                    coin_counter+=1
```

Complex code is interpreted for the reader always. The variables used are also unique, appropriate, and easy to read. The code is also formatted similarly with _ as a space.

```
#for loop to loop over the snowflakes and draw them randomly onto the
screen, this is convenient because they're so small and so many

    for i in range(2, 15):

        obstacles[i].set_radius(0.2) #life 3

        obstacles[i].set_position(rand.randint(0,89), rand.randint(24,
89))

        obstacles[i].set_color((255,255,255))
```

I use for loops, if statements and while loops where possible to make the code concise and precise.

5. **Provide plots, graphs, visualizations, screenshots and/or video links that demonstrate key parts or features of your project with descriptions of their significance.**

This is included as an attached video titled “final_project”.

You can see the lives disappearing in the top right, the coins being counted in the top left, the obstacle course being reset when the user gets to the end of the screen, and the user interaction as I choose the color and use the car to navigate the course.

6. **Identify any potential biases or limitations in the dataset(s) or solutions/models and what might be done to safeguard against any harm or negative impact of the analysis.**

I feel like Zelle Graphics isn’t a very good graphics generator for games and so I would probably recommend using pygames instead.

The checkKey function also makes it so that the user can jump infinitely many times which basically makes the games obsolete but the fact that the user also has to collect coins counteracts that by giving them an incentive to stay grounded and dangerously close to the obstacles. But still, I’d much rather if they just couldn’t.

7. **Properly cite your sources of consultation or inspiration for this project (e.g. peer or expert consultation, images found on the internet, research publications, websites, tools, etc.).**

Professor Allen Harper

Libraries used: time, random, collision, graphicsPlus, math

The game was inspired by a game from google playstore called **Hill Climb Racing**

Personal Reflection:

1. Clearly identify your programming and analysis strengths, areas where you are still learning, and why you chose this topic/dataset for the interview to demonstrate your skills and knowledge.

I feel like I'm very comfortable with using OOD as a programming approach and am beginning to understand why it's a convenient fix to a lot of problems that I could have while coding.

I also think that I've become very good at using complex functions, using return values, if statements and calling functions inside functions. In general, this gives me a lot of flexibility in how I choose to solve a problem.

In general, I feel like I could code anything in Python now and have an understanding that is good enough to absorb more information quickly and learn more dramatically.

I do feel like I need to work on more recursion problems, that's why I didn't have recursions as a solution to anything that I coded. The reason why I chose this project is because it puts an emphasis on OOD and function complexity.

2. How does this project demonstrate how much you have learned about introductory CS concepts, good programming practices, and computational problem solving?

The fact that I used modular design to code this project shows how much I've learned about introductory CS concepts. Each function is simple in of itself and then the way that they come together and work is complex.
I make an active effort to use good programming practices and make sure that the code is easy to maintain so that it doesn't get easily replaced by a better version of it. I also try to push my limits and do as much as I can rather than leave it to future iterations so I feel like CS has also given me an important lesson in being relentless.

I also make sure that someone reading the code can easily understand it and add as many comments as I feel like it would take to bring someone reading my code onto the same page.