CS-500 Module Three Journal

Isaac Raymond

CS-500-11462-M01

Professor Ronak Nouri

Modular Programming

The majority of modern world programs can and are being written as modular programs, and for good reason. We need to be able to quickly create our programs and we also need to be able to read and understand those programs. One such application would be in the development of a modern video game. There are many parts to a video game program. Video game programs often break. Reading, understanding, and fixing the code is much easier if it is built in a modular fashion. Teams can also work on game code much easier if it is modular, as they can quickly isolate a graphics problem by looking at the functions involved with the graphics rendering.

I would implement a modular program in a video game solution in the following manner: I would first create the game loop (a loop that runs until the user elects to exit the game). Inside that loop, I would have several functions. One function would be to implement any user input. The next function might be to calculate any physics calculations. The next function would render the graphics based on the results of those physics calculations. Inside each of these functions might be many more functions that break down the graphics calculations into smaller pieces, the physics calculations into smaller pieces, etc.

Here's an example game loop function:

```
def main_loop:
    interpret_user_input()
    list_ai_locations = implement_ai_character_inputs()
    make_physics_calculations()
    make_graphics_calclations
```

The implement_ai_character_inputs function might look something like this returning the location of each ai non-playable character:

```
def implement_ai_character_inputs():
    ... perform some randomization for direction of ai movement
    return list_of_ai_locations
```

The implement_ai_character_inputs function returns a list for further physics and graphics calculations.

The game would be almost impossible to expand if its code were not modular. A team of researchers would spend most of their time trying to just locate a place in their code they want to edit if they wanted to add new items. A file with thousands and thousands of lines of code is almost impossible to read and work through in a practical amount of time. Modular programming makes fragments of the code reusable, thus avoiding the copying and pasting of several repeating elements throughout the code.

Modular programming makes code much more easy to read in this context because a team member who hasn't worked on a certain area of the game can look at the function that's relevant to their issue and update that function accordingly. Without the functions and separation of code, the user would spend too much time sorting through code to try to find the line they want to edit or the area they want to add to.

The same graphics processing code might be used on each game object. A video game's file would have much too many lines of code if we only copy and pasted this particular line!

Instead, creating functions and using a modular design makes that one bit of cod reusable in other places.

When something goes wrong, a properly written modular code can easily be read to find the area that causes the issue. If, in the above example, there's an issue with the physics engine, a programmer can start by looking in the make_physics_calculations function to find the problem. Modular design makes testing and debugging much easier and quicker. And, whenever the code needs to be updated due to technology updates that the code is dependent on, the modular code lets the programmer quickly find and update that area of code.

Compared to non-modular coding, the benefits far outweigh any challenges. Modular coding has a challenge in that each function and section of code must be created with an organization system in place. Each organization usually has their own naming conventions, requirements for new functions, for new organizations, etc. These decisions are often judgment calls, but if an organization has a clear policy, a coder can follow it so that other programmers can read and update their code easily.

Modular programming is essential for the reasons listed above. Programmers might have been able to get away with one long string of codes in the very early days, but now, programs are very complicated and can't practically be completed without a modular design. Modular programming makes a coding project quicker, easier to interpret, and easier to fix and maintain.