Thomas Mathew
Final Project Design and Reflections Document -CS 162

Design:

I loved the flexibility that this assignment gave us! I considered a number of different game "themes", including a football player trying to run past tacklers, and a person trying to bake a cake successfully.

I eventually settled on a more game-like theme: Sonic the Hedgehog. It fit naturally with the requirements of the program: each "Space" would correspond to a Zone/Level that Sonic would "Run" through. I decided that using elements common to the Sonic games would further fit with the assignment.

Specifically, the player would be tasked with finding 5 different Chaos Emeralds, scattered across the stages. Along the way, they can collect Rings (protection from damage), collect extra Lives, and/or take damage (losing Rings or Lives in the process). If the player runs out of lives, the game ends (thus creating a sort of clock on the player-they must find all the emeralds before they run out of lives)

I created 5 different types of Zone for each of the required 5 spaces. I figured it would not make sense to have 2 different Zones be similar- in the Sonic games, you are running along Green Hills one moment, and underwater the next.

Function-wise, each Zone has:
-A constructor that simply initialized its name as a string. This string is used in display to the user.
-An "exitZone" function. This function is entirely inherited from the base "Zone" class, unlike the other functions, which re-define the body of those functions. "exitZone" demonstrates the linkage between the Zones via pointers.
-A simple "Welcome" function that gives the user information about the Zone.
-A "Special" function. Each zone has specific functionality, fitting the "theme" of the stage. For example, for the water Zone, there is a chance that Sonic has to surface for air, and thus doesn't proceed to search the zone.
-A "routeRun" function. The player chooses which path Sonic will take through the level, as in the actual Sonic games. The player may find an Emerald, Rings, be damaged, or find nothing at all.

My Main function asks the user for input to choose a Zone to start off in. From that point, the Zones take over: Welcome will run, then Special, then routeRun, and finally exitZone will get user input for the next Zone destination. Meanwhile, Main takes the results from routeRun (and Special in some cases) and translates it to changes in Sonic's inventory.

Testing and resolved problems:

Initially most of my problems derived from the way I had some of my functions set up. One issue in particular came up when I tried to pass parameters to the routeRun functions in my various zones (I wanted to pass in whether Sonic currently had rings, and how many lives he currently had). This caused problems with the inheritance from the base Zone class. I eventually realized that my routeRun functions didn't need parameters to operate.

Instead, I set up my main to interpret a wide variety of "results". To streamline, I had all results simple be of the form "int", and had all my relevant functions return an int based on the user's choice and result. For example, if the path chosen causes Sonic to get Rings, then the function returns 1. Then, main checks "result", sees that it equals 1, and adds a ring to Sonic's inventory.

I was concerned with getting the difficulty level correct – I didn't want to player to lose all their lives easily, but I also needed that to be a real possibility. My best way of doing this was to "play dumb" as best I could while testing the game (since I obviously knew what answers would lead to what outcomes)

While adopting this "play-dumb" mentality I realized that my program was not giving the user nearly enough information to understand the game and measure their progress. For example, I didn't have an explanation of what Rings do, or that different Zones would have different characteristics. I just assumed (implicitly) that the player knew who Sonic was, and how a Sonic game worked.

A problem I ran into later that that the way I was keeping track of the user's collected Emeralds. I was just adding up the collected emeralds, rather than making sure that the user was actually finding the 5 different emeralds, rather than just returning to the same stage and getting the same emerald over and over. So I built in a small bit of code in main to catch the case where Sonic already has the emerald of a given stage.

Another issue came in taking user input for the next zone that they wanted to go to. The program seemed to randomly perform correctly sometimes (going through the entire "while" loop in main), but sometimes it would just go straight to running the functions in the next zone. I eventually realized that I needed to place "cin >> zoneChoice;" separate in order to make sure the while loop ran every time.

In testing my game, I made sure to run the program over and over to catch any errors. I would enter zones randomly, rather than going into the first one first, the second one second, etc. The key was to test as many possibilities as possible in order to find all the bugs.
Some errors I found included:
-Collecting the same emeralds more than once (as described above)
-Not checking correctly whether Sonic had rings to protect him from damage
-Incorrectly using random numbers in the Casino Zone
-Not saving the result of the second attempt in the Green Hill zone

Reflections:

One takeaway was the necessity of putting myself into the user's shoes. Looking back, I realize that for many of our assignments, I implicitly assumed that the graded had a pretty good idea of what the program would do and how to work with it. But for an assignment this open-ended, I was forced to consider more strongly how much information needed to be presented to the user in order for it to be accessible.

I was surprised by how fun the assignment was – normally when I think of a final project I think of toiling away endlessly on something that I just wanted to be finished and over with. But in this case it was nice to be able to choose my own theme, and it sparked some creativity in terms of setting up the Zones, etc that the player interacts with.