Comparison with Original Design Specifications

My game has mostly followed the artistic style and gameplay mechanics outlined in the original design specifications, but has diverged in the game content and levels due to the time constraint and my underestimation of the difficulty of implementing some of the features.

My game does not have any pre-made levels as depicted in the original design specifications due to the time constraint of the major project. In order to add pre-made levels to the game, I would have to design a system for me to edit, serialise and load them. I underestimated the time it would take to develop the procedural game mode, which left too little time to implement these required aspects.

As well as this, I have re-envisioned the procedural "infinite" game mode depicted in the original design specifications. This is now the only game mode, and is not infinite but instead the player can choose to play either 5 levels or 10 levels in either easy or hard mode. I have kept the main feature in the design specifications for this game mode which is that the levels are randomly procedurally generated and so are unique each time.

Logging Tests

I tested the initial versions of the procedural cave generation algorithm by running a file that called the function generating a cave (the initial algorithm simply returns a 2d array with 1s and 0s for walls and empty spaces), and then passed the generated cave array to a function that printed it out. One of the issues that I came across was that the cave would output either completely empty or completely filled in. Fixing this required tweaking the cellular automata algorithm and the parameters passed in.

Test camera module by resizing the window and ensuring the view is scaled correctly and the aspect is not distorted.	Works as expected
Testing Main Menu Screen: click on Play button	Works as expected (goes to choose mode screen) in <100ms
Testing Main Menu Screen: click on Quit button	Works as expected (quits program)
Testing Main Menu Screen: click on Sound button	Works as expected (toggles sound on/off)
Testing Main Menu Screen: navigating to screen with sound on/off: the toggle sound button should reflect whether the sound is	Works as expected

on/off when the screen is loaded	
Testing Main Menu Screen: click on Help button	Works as expected (goes to help screen)
Testing Help Screen: click on back button	Works as expected (goes to main menu screen)
Testing Help Screen: navigating to screen with sound on/off: the toggle sound button should reflect whether the sound is on/off when the screen is loaded	Works as expected
Testing Choose Mode Screen: Clicking each option sets the right difficulty/levels	Works as expected
Testing Choose Mode Screen: navigating to screen with sound on/off: the toggle sound button should reflect whether the sound is on/off when the screen is loaded	Works as expected
Testing Choose Mode Screen: click on back button	Works as expected
Testing Play Screen: drag back should create preview for shot (test all directions)	Works well
Testing mouse behaviour when making shots	Using a mouse it works perfectly Using the macbook trackpad, the mouse release event gets dropped often. Researching, I am almost certain this is an issue with either the trackpad or pyglet and have found an open issue on the pyglet repository of the same issue being replicated
Press “G” to enter/exit goo mode	Works as expected
Pressing “G” mid-shot should cancel the shot	Yes
Hovering over contours in goo mode should show a light preview for where the goo will be placed	Yes
Test that the goo is joined together (this is important as the edges are rounded) when one is placed adjacent to another	Yes Tested this both in playtesting and also in the code by printing out the goo array to ensure that two goo classes are merged

	into one when they are adjacent
Goo can be placed (and not on flat ground/in sand pits)	Yes
The ball sticks to goo	Yes
The ball can be shot away from the goo	Yes
The ball sticks to sand pits but not if lots of horizontal velocity	Works decently well
Level count/shot count works as expected	Yes
Test clicking pause button	On my MacBook the blur works, but on my iMac the blue effect doesn't work (where the background is blurred) Other than that it is functional and the buttons to start new game/go back to main menu work
Ball in hole updates score gui and generates+starts new level	Yes
Final score count (number of shots) is correct	Yes
Play again/back to main menu buttons work	Yes
Clicking button sound effects work	Yes
Ball hit goo sound effects work	Yes
Ball hit sand sound effects work	Initially if the ball slid against the sand, the sound effect would be played every update leading to a constant, unpleasant screech. To fix this I require a minimum vertical velocity for the sound effect to be played, mimicking a plop sound when the ball falls down instead of just travelling sideways
Shot sound effect works	Yes
Ball in hole sound effect works	Yes
Randomly generated levels are all playable with a path from start to hole	Yes

The majority of my testing during the development of my game was program testing as I implemented new features. My development approach is the evolving prototype, and so as I implemented a new feature, I would run the entire game and test the feature by accessing it in the game. If it didn't work as intended, or I came across an error, then I would go back into the code to scan if there was anything that stood out as being wrong. Then depending on the error/misbehaviour I would try to identify where in the code the error was originating from. To do this I printed out variables to see if their values were as expected, commented out parts of the code to see if they were causing the error (if the error persisted despite their omission), and sometimes would resort to unit/module testing the new piece of code in isolation, with controlled inputs to diagnose what was going wrong with the code. For example, after writing all of the tesselation code that generated the geometry to render the cave walls and ground, I ran the game and it immediately crashed with a segfault. I determined that the segfault was occurring inside a double loop with a bunch of OpenGL calls by commenting out different parts of the code. Then by analysing the code I realised that the issue was caused by the fact that I was allocating the geometry inside the loop, then was using it again outside of the loop causing a segfault. To fix this I allocated the geometry outside the loop and stored a reference to it until I was done using it.

Another issue that I found when program testing was that the text in the GUI would not be rendered occasionally. It seems like when this happens is random, and I believe that it is an issue with either pyglet or the glooey library as I have checked that the fonts are being properly loaded. Navigating to another screen and back leads to the text showing as it should.

Tested the program on my MacBook - macOS Monterey - MacBook Air (Retina, 13-inch, 2018) - Processor 1.6 GHz Dual-Core Intel i5 - Memory 8 GB 2133 MHz LPDDR3 - Graphics Intel UHD Graphics 617 1536 MB	The program would crash with an error where it could not properly access ffmpeg's dependencies. My program doesn't use ffmpeg, but it is loaded by the pyglet library. So, to fix this error, I modified the pyglet install, deleting where it loads ffmpeg. After fixing this the program worked.
Tested the program on my iMac	The dynamic blur effect in the pause and end game screen does not work on the iMac due to an OpenGL behaviour difference. Everything else works