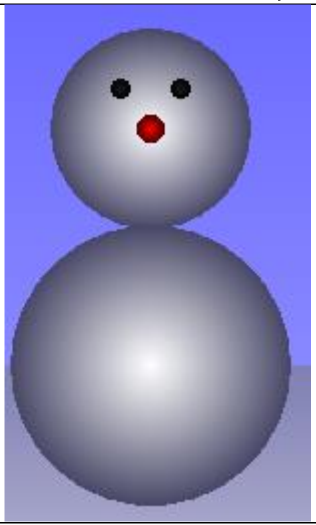
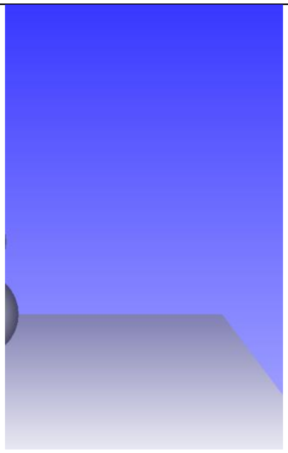
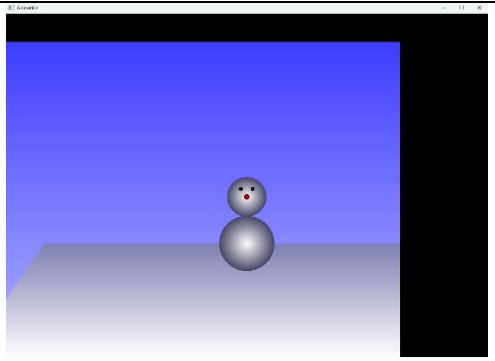
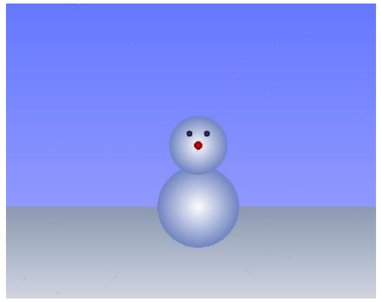
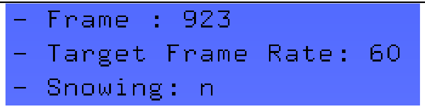
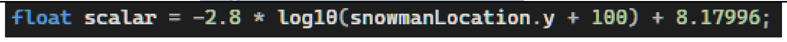
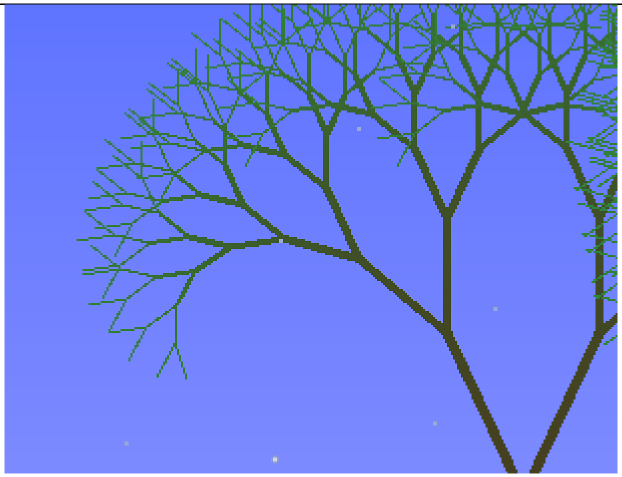
Robin Nowlan

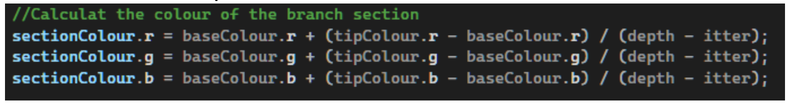
Time Spent	Date	To do	Work done	Detail	Bugs
2hrs	23/07/24	Define basic structs	Implemented the beginnings of my addon library (GLUE)	For this project I am making a library with all of my most used functions to aid in the development of the project. I am calling this GLUE (GL Utility Extra). This allows for abstracted processes and cleaner code. I am using two structs, first is colour which combines the RGB components into one strut. The next is similar as it is location, which combines x and y components.	.
3hr	23/07/24	Draw shaded circles	Created functions to draw circles that taper off their colour to the edges to give the appearance that they are 3D	 The circles are created with a triangle strip which has its first vertex in the centre. This is set to the highlight colour (white) and all the vertices on the edged are set to the falloff colour (in this case dark blue to replicate global illumination from the skybox).	The colour shading only allows for a linear transition of colours so the shading does not look natural as real 3D has cubic fallow (inverse square law)

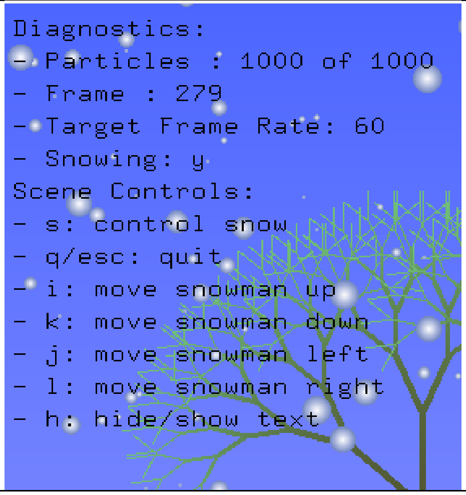
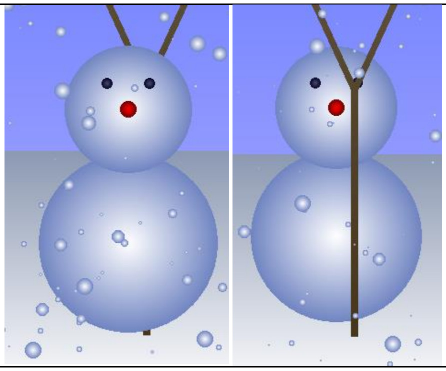
1hr	23/07/24	Draw sky and floor	Used simple QUADs to draw a rectangle with a gradient as the background. As well as use a mis-shaped quad to simulate the floor disappearing in a 3d way.	The background fades lighter as it gets further down the screen, and the floor does similar but gets darker as it goes back to make it look like it is going backwards. 	 The floor and snowman use the relative size of the screen to adjust the scale to the size of the window. However despite being set the same way the background does not scale beyond a set size, instead just turning black.
1hr	24/7/24	Modify colours to look more natural	I spent a while finding colours to make the scene look more natural. To do this I also had to implement a colour tint for shading.	The snow man now has blue tinted shadows to reflect the blue scene 	-

3hr	24/7/24	Implement text	Created a function that can show any string at any location	I also added a way of converting numbers (floats and integers) as string numbers to display stats  `- Frame : 923`
				`- Target Frame Rate: 60`
				`- Snowing: n`
					-

5hr	25/7/24	Get snow working		Created a particle struct in the GLUE framework. Created an algorithm that randomly generates snow and snow velocity. Snow velocity and size are linked to give the perspective that there is depth. Particles are regenerated after they fall off screen. 	I had a bug where the snow would all form at the top of the screen and would fall down in one big curtain. This was caused by the snow being generated with the same Y height of the screen height. This was fixed by generating the snow height to be greater than the screen size. Effectively setting them above the screen height and letting them fall into frame. Giving a more realistic snow fall.
3hrs	2/8/24	Moving the snow man	Using the relative screen position to move the snow man around the scene depending on the users' inputs	`float scalar = -2.8 * log10(snowmanLocation.y + 100) + 8.17996;` This was a challenging part of the development as the snow man could easily just move left and right as the X axis does not need to change the scale of the snowman, however when the snow man moves in the Y axis (false Z axis) it needs to look as though it is getting larger and smaller as it gets closer and further away. At first, I tried using a linear relationship but that does not look natural. Eventually I ended up using Desmos to plot out a graph representation and found that a log function has a good look to it. Using Desmos I found the coefficients that worked best for my scene. Finalizing on the function: $scale = -2.8\log(y + 100) + 8.17996$	There was an issue where the eyes of the snowman would drift further and closer apart depending on what the y position is. I found that this was caused by the eye position not being scaled by the scaler. This was an easy fix.
2hrs	6/8/24	Using gaussian noise to generate mountains	I was looking at using recursion and gaussian noise to create a background of mountains	This was an experiment that I spent a few hours writing the framework for but ended up not going with. The idea was to generate unique mountains with a 3D low poly feel which would gather snow as it is snowing and the snow would slowly melt over time as it stops snowing. Eventually I decided against this idea as I thought of another feature to implement.	-

4.5hrs	8/8/24	Recursively generating fractal trees	Inspired by that weeks lecture on fractals I wanted to generate trees which like the mountains would gather snow as the snow falls and turn back to green when it stops snowing	 To draw these trees I used two functions the first takes in the location, colour and branch depth that wants to be drawn. This begins the recursive process where each branch is drawn. To be able to draw the branches I first had to implement a function that could draw a line of a given thickness between two points. This was done using basic vector math. From there each branch is drawn using the end point of the previous with a given length and angle. I found that an angle of 20 degrees looks best for this application.	I had some issues implementing this as I was working in degrees but the trig functions in C work in radians. This was an easy fix but it took a while to figure out what was going on.

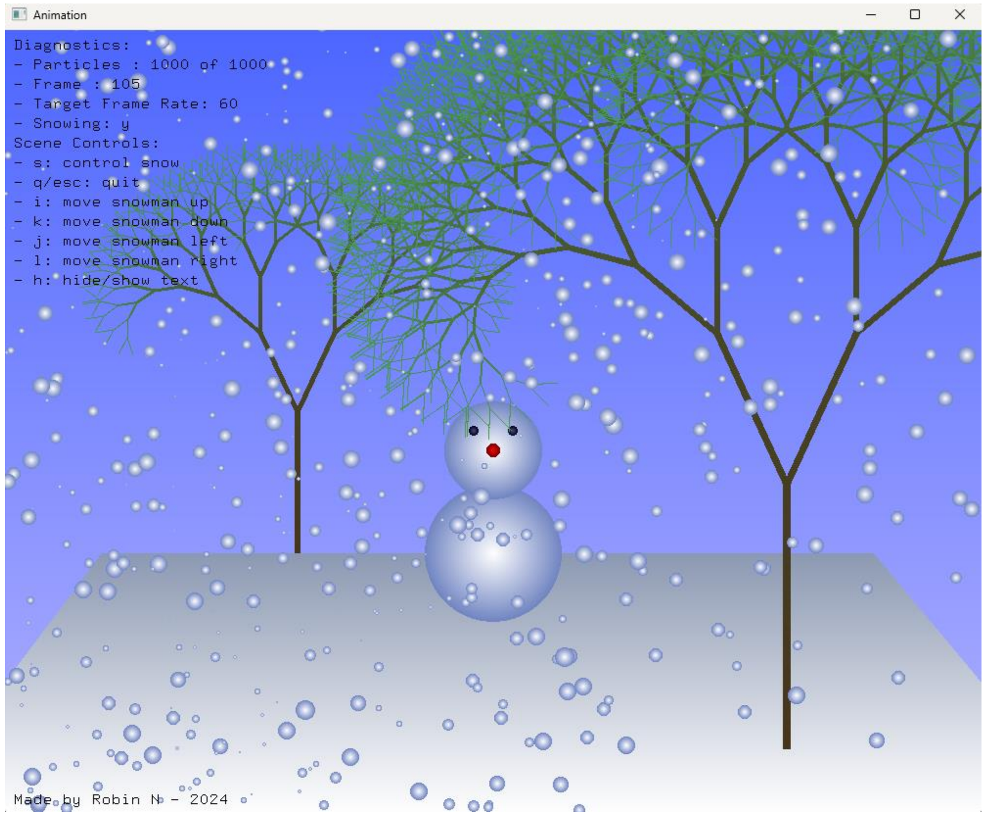
2hrs	9/8/24	Tree colour blending and snow fall colouring.	Allow the tree to blend from one colour at the bass to another at the tip of then "leaves"	This was complicated to implement within the recursive structure that I had created. The solution was to have the colour of the end of the previous branch as the base of the next branch.  `//Calculat the colour of the branch section`
				`sectionColour.r = baseColour.r + (tipColour.r - baseColour.r) / (depth - itter);`
				`sectionColour.g = baseColour.g + (tipColour.g - baseColour.g) / (depth - itter);`
				`sectionColour.b = baseColour.b + (tipColour.b - baseColour.b) / (depth - itter);`
				Using the above formulas, I was able to scale the RGB values of the colour to fade between to colours. From this it was easy to implement the idea of snow falling on the tress. Each frame shifts the tip colour by 0.01 (shifting the colours lighter makes it look white). The shift is positive if it is currently snowing, negative if it is not snowing. This gives the illusion of the snow melting or building up on the trees.
					One issue I had with this was when it was not snowing the colour of the tip would fall past the green default colour and go black. This was due to there not being a lower bound on the colour. I fixed this with a simple if statement which clamps the colour to between the green leaves and the white snow. Ensuring that when there is no snow whatsoever the tree is nicely green, not black.

1hr	13/8/2 4	Full text menu	Add all of the diagnostic text and keyboard controls	 Diagnostics: - Particles : 1000 of 1000 - Frame : 279 - Target Frame Rate: 60 - Snowing: y Scene Controls: - s: control snow - q/esc: quit - i: move snowman up - k: move snowman down - j: move snowman left - l: move snowman right - h: hide/show text	This was easy to implement using the text function I had already created. I just offset each line of text by 20px. I also included a show/hide keyboard shortcut that allows the user to hide the text (all text is hidden except key to re show the text).	-
1hr	14/8/2 4	Limit snow man movement	Prevents the snowman from moving off the "floor"	This was a quick implementation which ensures the snowman offset cannot allow the snowman to go off the floor or come to close to the "front" of the screen to cause issues (the log function could mean the snowman covers the entire screen).		-
0.5hr	14/8/2 4	Depth mask for trees	Draws the snowman either in front or behind the tree depending on its y position.		This was a simple check to see if the snowman is behind a limit or in front. Behind the tree is drawn first, otherwise the snowman is drawn infront. This means renforces the illusion of depth.	-
1hr	16/8/2 4	Cleaning up the code	Removing unused functions and variables	Throughout the coding process there were a lot of unused code which was left behind. I spent an hour going through and removing variables and functions that have no use. As well as this I found a lot of magic numbers that should be declared, so I		-

			as well as magic numbers	created some macros or variables to store these to make the code easier to read and modify. There are still some magic numbers however I deem them acceptable without a lot more work and in some cases a complete restructure of the code.	
Total =30hrs					

Conclusion

Overall, I found this project to be a success, given more time I would of like to of implement the mountain idea that I had. Despite this I managed to create a really effective framework for any future modification of this project. I believe the code I have written makes sense, if efficient and is easily modifiable. In terms of graphics, I like the look of the scene, especially the trees. Shading would of made the scene look better however this was outside the scope of the project.



```
Diagnostics:
- Particles : 1000 of 1000
- Frame : 105
- Target Frame Rate: 60
- Snowing: y
Scene Controls:
- s: control snow
- q/esc: quit
- i: move snowman up
- k: move snowman down
- j: move snowman left
- l: move snowman right
- h: hide/show text
```

```
Made by Robin N - 2024
```