Derek Rivera
Chase Kimball
Professor Weiss

Project 2 Frogger

In the beginning of making the code we would have to determine what we wanted our cars and frog to look like. So we went with the cars that looked like simple rectangles and the frog as a triangle. This design choice is because we wanted more time to code the logic behind the game and it is simple to keep track of the cars and the frog.

Then we would use some of the code from the gaskets to make the cars move on their own such as xLoc = gl.getUniformLocation(program, "xShift");. and some of the code from our house homework so that we can make the frog move independently from the cars such as gl.uniform1f(xLoc, 0);. This would ensure that the frog will not have the same xShift as the cars would. Because when developing that portion of the code we would be able to move the frog but the problem was that the frog would move side to side with the cars.

For our cars we would make it so that they would be able to move by using a 2d for loop, this will ensure us that each of the cars would move to the side according. Then another thing that we would do is to make the frog move by using a switch case to make it much more efficient than an if statement. Then we would also make the stop and start buttons so that we can start the game whenever the user is ready and then stop whenever the user would need a break. Next we would then make the general stuff for the viewport and clear color. But we would then learn about the binding of buffers. We can make different buffers according to what set of coordinates we would have and what we want to do to them. That is when we would have the regular buffer that was reserved for the cars and the frogBufferId would be reserved for the coordinates of the frog. We would also add an mp3 sound at the end to make like a playful effect as you will see in the future code we will have that in different occasions, and we would have specific function for the audio cues.

Now to show what the xShifts is all about we would have to start with what the var xShifts would mean. Var xShifts = new Array(9).fill(0.0) is a way of telling the computer that there will 9 slots in the array (which will be for the 9 cars in the game) and the fill part of the xShifts is that it will automatically fill the slots with 0.0 because we aren't using them yet.

We would also have a showMessageAndReset function where as the name says it would show the gameover message and then restart the game. But we would also use a variable called gameState which will notify the code if it is still running and keep rendering or if it will need to start the game over. In order to do this we would have the parameters of message, newState, resetSpeed. Now the newState will indicate what type of phase the game is in, the message is what type of message we would display on the screen, and finally the resetSpeed is where we would have that pause of the game to not enter the phase immediately

Now in the render function we would have to first check if the gameState is at “playing” mode so that it can execute the rest of the code, then that is were we will start to make the game and the drawing of the cars. Now as you probably saw the cars would be having different random colors and this can be seen in the first part of the code by using Math.random(). For the drawing of the cars we would use a 2d loop so that we can be able to set the y and x coordinates of each of the cars, as seen in the code there will be xShifts[carIndex] so that we can put the coordinates of each of the cars there, but you can also so there are three variables that will affect this. The first that i want to talk about the is the direction which is controlled by the stop or start button which is very important because it can either be a 1 or a 0. The 0 will just make the whole array of the cars not move at all while the 1 will allow it to move, now for the directionMultiplier this how the code would invert the direction of the cars based on which row it is on. So if it is on the 2 row it will tell the code to make those cars to go from right to left and then if it is on a different row then it would allow it to invert once again, then finally the last variable is the code that controls the speed of the cars which is the carSpeedMultiplier.

We would utilize in previous codes of the idea of xShifts but with a little twist. When it reaches the end of the screen it would make it end up on the other side of the screen once again. Then we would also involve the u_translation to help aid with the movement of the cars. We would also involve the uColor from the HTML where it can be linked to the color of the cars. Then we would include a good tool which is gl.flush() which will allow us to get rid of the buffers that we would currently have.

Now on to the frog section of the code, we would again use the 2d loop so that we can get the index of the cars and use it to also look at the indexes of the frog. Now the main thing that is giong on is that before drawing the frog we would have to make sure none of parts of the frog would be touching the cars. So in order to do this we would have to first have the right mathematics to have specific spots of the cars in working order then with the Split function we could make some shallow copies of the different cars. Now each car is made of 12 total points, 6 being triangles. So the problem we would have before is that we were only keep track of the triangles that make up the car istead of the whole it self, which is what was seen in the demo where the frog would get killed by phantom cars.

Now for the next code we would use the spread operator to basically traverse the array and then with the aid of .filter() we can pass through without any repetitive code or altering the array. Another great tool that was used to find the specific locations of the cars is the idx which is basically used to travel through the nested part of the arrays. After making this part of the code we can now sort through every car that is displayed on the canvas and know exact where their position is at every given moment.

With that piece of code established we would be able to do some simple logic with our collisions which would give use a boolean variable and depending on that variable we can say that the game state is is at a game over, then pass the mp3 sound of game over and then finally reset the game

Then at the very end of our code we would pass the frog array into the buffer and link u_translation to the frog so that it can move and then with gl.uniform1f(xLoc, 0.0); we can make the frog move independently from the cars and finally we would give the frog it's natural color of green.

Enjoy the frogger game!