AP report

Stone Implementation

For the implementation of the stone class, it has to control the velocity and team of the stone. To implement this, it has a position and velocity to control the movement of the stone. These are implemented using the update function where the velocity decays based on the friction force and the position is updated with this new velocity. To start the stone moving it will use the ApplyImpulse() function which will set the velocity to the whatever the impulse is. This will then cause the update function to start moving the stone at the speed of the impulse with the decay of the speed with the friction force. This friction force is calculated by the ApplyFrictionForce() function which takes the current velocity and decays it using the current velocity as the negative acceleration. This causes the deceleration of the stone to decrease as the stone slows which adds the feeling of it sliding over the ice.

Sheet Implementation

For the implementation of the sheets I created a class which would then be used to create a single sheet with a variable called sheetNum which would hold the number of the sheet allowing it to be easily placed in a different location. To move the sheet I used the sheetNum with an equation of where x is the sheetNum. This equation means that when x is 0 the sheet gets placed in the middle position with the sheets then alternating left and right whilst moving out. This means that the sheet 0 will always be in the middle of the layout. The sheet consists of the 4 main walls around the outside which bounce the stones off them, the hog and hack lines which show where the stones are removed if they are not in between and the circles which make up the house. The class also stores the current stone count for easier loping through all the stones, a vector of the stones to allow pushback onto the end of the vector, the cushion and feature arrays which hold the graphics and information about all the drawn objects other than the stones, a particle set which holds the information about the particles, the hog and hack line positions and the centre of the house position. The cushion array holds 4 cushion class objects which contain all the information about the cushions for the game to draw and interact with them. The feature array holds a mixture of line and ring class objects both of which are children of the feature class. These arrays are used to draw the features of the sheet such as the hog and hack lines and the rings on the house. The hog and hack line positions are integer values which are used to test against the vector positions of the stones to see if an of them need to be removed. The centre of the house is a 2d vector which stores the position of the centre of the house which is used to score the stones at the end of a round.

To create the cushions array the class uses the function SetupEdges() which manually sets all of the cushion dimensions and adds them to the cushions array. It also sets-up the centre and normal of all the cushions for the bounce of the stones. The SetupFeatures() sets-up all the extra visual features of the sheet including the hog and hack lines as well as storing their positions in the relevant variables and the rings for the house including storing the centre of the house in it’s variable. The AnyBallsMoving() function is a function to check if all the balls have stopped moving by looping through the balls vector and checking the velocities of each. This allows the main loop to easily check when the next stone should be spawned, when to check if the stones are past the hog and hack lines and when to allow the player control of the stone again. The CheckStones() function loops through the stones and removes any that are beyond the hog an hack lines. By looping trough from the last element this avoids the issue of it accessing an element that has been moved due to the removal of another stone or it skipping a stone that is beyond the lines and should be removed. The AddStone() function simply adds a stone to the starting position of the sheet and appends it to the vector of the stones, also incrementing the stone count by 1 to reflect the new count of stones. The GetScores() function is used to produce the scores of the sheet at the end of a round it uses the scoreCentre variable to calculate the distances of each of the stones and see if they are on the house. The update() function is used to check the position of the particles to see if any need to be removed from the scene to reduce the load on the computer of rendering many particles even if the camera can see them.

Comparison

I have developed this game using C++ however I could have used another programming language such as C# or python. Python is a much higher-level language than C++ being more abstract and often easier to read. This makes it much better for entry level and beginner programmers but can sacrifice useful features and options (Ateeq, Habib, Umer, & Muzammil, 2014). The loss of features means that it can be slower than C++ in many instances (Prechelt, 2000), especially in parallel instances, as the python GIL (global interpreter lock). The GIL prevents multiple threads from executing python bytecode in parallel which is required as Cpython’s memory management isn’t thread safe. Cpython can be multithreaded but only 1 thread can be executed at once which reduces the usefulness of multithreading. Some processes happen outside of the GIL and can utilise multithreading such as disk or network I/O and so multithreading is only a bottle neck for programs that have much of their execution time inside of the GIL. In comparison C++ allows full multithreading of the program relying on the developer to manage the threads and any race errors. It gives you many different methods to prevent race conditions and problems with shared data such as atomic functions and barriers.

# Bibliography

Ateeq, M., Habib, H., Umer, A., & Muzammil, R. U. (2014). C++ or Python? Which One to Begin with: A Learner's Perspective. *2014 International Conference on Teaching and Learning in Computing and Engineering.* Kuching, Malaysia: IEEE.

Prechelt, L. (2000). *An Empirical Comparison of C, C++, Java, Perl, Python, Rexx and Tcl for a shearch/string-processing program.* IEEE Computer.