Project Title: Motion Processing with ExMoveGraph

1) Introduction - MoveGraph with Keyboard Control

In this Project, a skeleton (character) made up of many joints has been used to examine the concatenation of motion processing. The main goal was to make the character move across the obstacles placed on the floor in a perfect manner. Objects of different shapes and sizes are placed on the floor as seen in Figure 1. To move the character, the keyboard keys have been used. Experiments with Blending function have also been done in order to make the transition between the motions in a smooth manner and prevent jerks during the change of states.

2) Design and Methodology

The Exmovegraph source code has been used to implement the motion graph algorithm. Initially, an in-depth study of the source code was done to understand the motion processing. To add new motions to the graph, .bvh files have been added such as turn180_mirror.bvh. To change the direction in which the character is moving and to achieve the required angle of rotation, the duration of the motion has been reduced so that the character does not complete the total 180-degree rotation. This also helps in modifying the global root rotation. Additionally, to make the character not collide with the objects placed in the motion path, Collision Detection is done by calculating the distance between the object and the current position vector of the character. The collision detection is implemented only on some primitive in the project. The 4 boxes on the corners and the 2 long bars were able to detect the character and make it move in the opposite direction.

3) Implementation challenges

During the implementation of the algorithm, a number of challenges were seen that were not trivial to solve. Some such challenges will be discussed in this section. Firstly, as soon as the character changes its state i.e. on pressing a key, the updated motion takes some time to activate it-self which eventually causes a delay in the motion graph. Due to this delay in activation of new state, motions were not being implemented immediately at the press of a key as seen in figure 2. Secondly, during the collision detection, sometimes the character tends to go inside an infinite loop which makes it to revolve around the detected object which can be seen in Figure 4 (Box placed in down right corner). Although this challenge is rarely seen, it was interesting to visualize it.

4) Result and Evaluations of the Proposed Solution

To solve the challenges discussed in the previous section, few strategies were used to solve them. To resolve the issue of the motion being implemented after few seconds, various things were tested. The distance travelled by the character after pressing the key was reduced by some specific amount so that the character covers less distance before it rotates by the required angle. Also, to solve the turning problem, motion was initiated at some frames ahead of the start motion and then that specific motion was applied at a small interval before the local time i.e. the previous local time (local time = local time - small interval). This helped to resolve the issue as seen in Figure 3.

Furthermore, to solve that challenge that was seen in the collision detection was partially solved by changing the position vector of the object. Rather than specifying the center point of the object, the coordinates of the edges of the object were calculated and set as the collision detection point. This helped to partially solve the challenges. After fixing the issue in Collision Detection, perfect results can be achieved as seen in figure 5. However, sometimes Collision detection does not work correctly while playing with the move graph. Furthermore, to optimize blending or to minimize the jerk at change of motion, variations were done to the blending window. It was seen that as the Blending window value was increased, the character behaved in a more rigid manner reducing the smoothness of the motion processing. Further evaluating the proposed solution, it is seen that the character is able to pass even through the narrow spaces only when the key is clicked at the right time.

5) Conclusion

In this project, various new topics were studied and evaluated. The results and figures show that the character is able to move across the obstacles in a smooth manner although some mis behavior is seen few times.

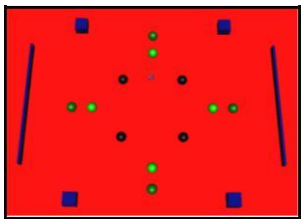

Figure 1: Floor with Obstacles

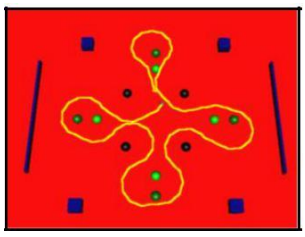

Figure 2: Before optimization

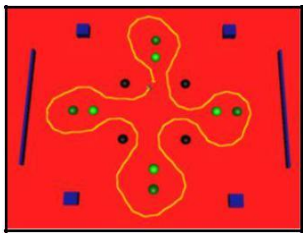

Figure 3: After Optimization

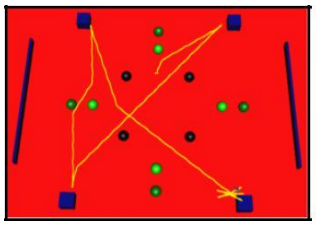

Figure 4: Character looped in Collision Detection

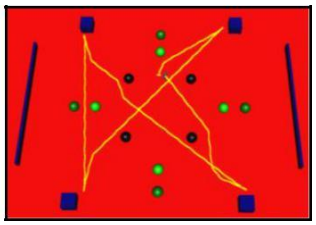

Figure 5: Perfect Collision Detection