Boids Algorithm for Bird Flocking Simulation

Anthony Bartholomew – bartholomew.122@osu.edu, Kyle Lambert – lambert.448@osu.edu

Abstract—This project aimed to simulate the flocking of birds using Boids algorithm. Boids algorithm uses three key properties that emulate flocking which are separation, cohesion, and alignment. By using Boids algorithm, the team strove to create a simulation that shows accurate movement of birds as they fly around the environment to create flocks with other birds. This project examines the computer graphics problem of how realistic behaviors observed in nature be simulated without introducing high levels of computational complexity. Using basic WebGL, the team found that it is possible to simulate the pattern of flocking among large amounts of birds without sacrificing performance. It was also possible to improve performance in the Boid algorithm by streamlining repeated work for each of the four key properties. The team also implemented a WebVR version of the simulation which wraps the environment around the user. Using a VR headset or Google Cardboard allows for a more immersive experience as the birds fly together.

Introduction & Motivation

For the project, the team thought that it would be an interesting idea to visualize realistic behavior of animals in nature using computer graphics. After doing some research, the team found Boids algorithm that was created by Craig Reynolds. The algorithm is a method for simulating flocking behavior used by objects called boids, which will be animals in nature in terms of the project. Flocking is the act of boids, or animals in this case, congregating together in a massive group as they move from one place to another. This behavior can be observed in animals such as birds. As birds are flying, they generally tend to move in the direction of other birds, thus forming large groups together. When birds are in these groups, they try to avoid crashing into one another, head in the same direction as the group, or break off and form smaller groups with other birds. Finding ways to improve algorithm performance and simulate realistic behaviors are both important to the future of the field of computer graphics.

Related Work

The team's research goals were mainly to find a way to implement Boids algorithm. Boids algorithm is able to replicate flocking behavior using three key principles, which are cohesion, separation, and alignment. Cohesion is the ability for boids to find the average position of its relative neighbors for the purposes of staying together with the group. Separation is the ability for the boids to find the average distance between itself and its neighbors so that it can avoid colliding into other boids. Finally, alignment is the ability for the boids to find the average direction of its neighbors so that it can fix its orientation and move in the same direction of the group.

Another goal was to provide both a virtual reality and non-virtual reality experience. As such, the team researched how to map the scene to a cylinder around the viewer.

After researching and implementing the Boid algorithm, the team wished to find ways to improve its performance as more objects are added to the environment. Improvements can be made by finding repeated blocks of code and calculations that can be made more efficient. When determining the effects of cohesion, separation, and alignment, some calculations were repeated for each boid in the scene. The team determined that by computing this information only once, performance could be improved significantly. Adding more boids to the scene allows for larger flocks of birds, leading to a more interesting experience for users.

Design Method

In terms of design considerations, the team initially decided between modeling birds or fish for the scene. Ultimately, it was decided that birds would be more appropriate for the flocking algorithm.

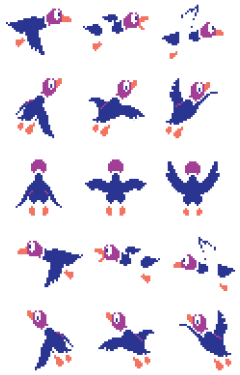

Figure 1

After finishing the initial implementation, the team decided to theme both the birds and clouds after the 1984 Nintendo Entertainment System game *Duck Hunt*. This required a custom animation system that utilized a texture atlas to switch between frames and animation sets. This system changes based upon the heading of the velocity vector. The heading determines the direction the bird is facing, which in turn determines which animation set to use. One texture atlas for a bird is shown above in *Figure 1*. The use of custom animations provide a more engaging and visually interesting experience for users.

Implementation Method

This project makes use of low-level WebGL 2.0, which uses HTML and Javascript, to create a 2D implementation of the Boids algorithm. Birds are represented in this environment using the graphics and animations from the game *Duck Hunt*. WebVR

is incorporated into the design to allow for a user to view the flock in virtual reality. The project is viewable on any Android device through the Chrome web browser, allowing for use of Google Cardboard or any other mobile VR headset. Users can add more birds to the scene by clicking anywhere or touching the screen.

Results & Analysis

As stated in the previous section, the team successfully implemented Boids algorithm utilizing a custom animation system so that boids could be textures using *Duck Hunt* animations. When the program is not in VR mode, the boids, or ducks, are draw onto the xy-plane. As a boid approaches the edge of the screen, it will fly out of view for a short distance and then wrap around the plane to the opposite side. In *Figure 1*, the program can be seen running in non-VR mode.

Figure 2

As it can be seen, the team also included clouds to fill in the background in order to make it make it look more sky-like. When the program is in VR mode, then the boids will have their planar positions mapped to a cylindrical plane that is centered at the origin. Additionally, the user will be placed at the origin of the scene so that they are inside of the cylindrical plane. In *Figure 2*, the program can be seen running in VR mode on the Chrome browser using the WebVR API Emulation extension, which is creating the scene twice for each eye.

Figure 3

Outside of the desktop emulation of the VR mode, in *Figure 3*, the program is running on a mobile android device using the Google Cardboard view.

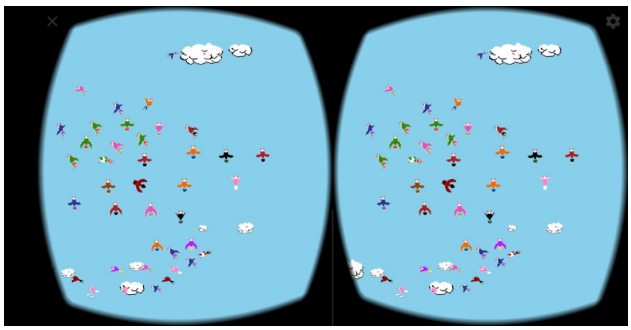

Figure 4

As the program is running on a mobile device, the user can physically move their head in the world to see how the boids are interacting with each other, which drastically increases the level of immersion for the user. One important note about mobile devices when it comes to this application is that iOS devices cannot run the program properly since WebGL 2.0 features are currently considered as experimental/developer only options.

Discussion of Results

In terms of performance, the team tested the number of frames per second output as the number of birds objects in the scene were increased. These tests were performed for both the VR and non-VR scenes. Clouds were also removed from this test as the differences were negligible.

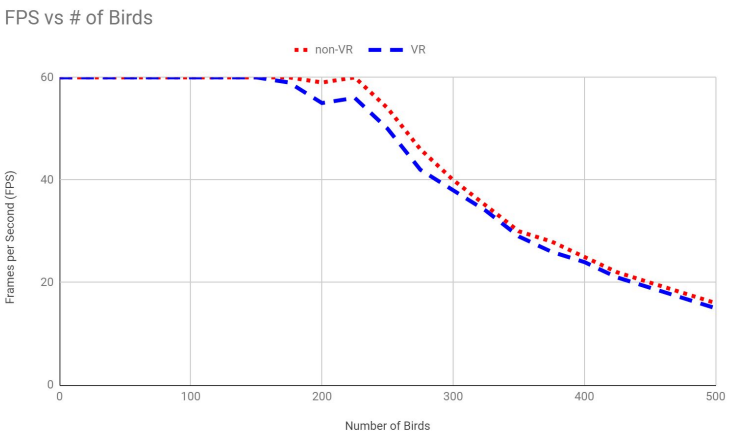

Figure 5

As shown above in *Figure 4*, the program maintains the maximum amount of performance at 60 fps up until around 225 birds. At his point, performance in both viewing modes starts to drop nearly linearly. There are a few dips which can be explained by changes in the amount of boids within a certain searching range of each other. The number of these calculations vary depending on the positions of each boid in the environment. Once reaching 500 boids, performance had dipped to less than 20 fps in both modes. The performance in virtual reality is very slightly worse as compared to non-VR. This is likely due to rendering the scene twice, once for each eye. Performance will also likely vary depending of the hardware used. These performance tests were done using a laptop with an Intel Core i7 2.80 GHz processor, 16GB DDR4 RAM, and a GeForce GTX 1060 with 6GB VRAM.

Conclusions & Future Work

Overall, the team gained experience in visualizing realistic behavior of birds using the famous Boids algorithm, insight into how to use low-level WebGL, and how to integrate WebVR into a WebGL application.

In the future, the team would be interested in increasing the performance of the application. Performance could be increased by implementing a spatial partitioning algorithm that breaks the area of the screen into “buckets” that the boids can be placed into and removed from. Once the boids have been placed into their respective “buckets”, then an individual boids would only have to look at other boids in the same “bucket” instead of looking at every boids per update as they do now. Additionally, performance could be increased by reducing the number of WebGL rendering calls by taking advantage of instance rendering. Currently in the code, each boids and cloud on screen has to be drawn individually, which means that the CPU must communicate to the GPU that it should render an object to the screen each time. In instance rendering, the communication overhead between the CPU and GPU is vastly reduced since instancing allows for the rendering multiple objects in a single draw call.

References

[1] M. Larson, S. Lundgren. Perception of realistic flocking behavior in the boid algorithm. Blenkinge Institute of Technology, October, 2017