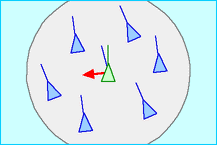
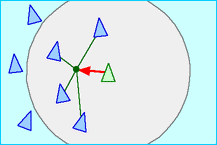
Role Specific Training II Game Development – Ruben Geerts – CMV2D  
  
Analysis  
This document will describe the process behind my role specific training assignment, regarding flocking. During class we also discussed Artificial Intelligence, but that was too hard for me to complete. I decided on this together with the programming teacher.   
  
The demo delivered with this document represents flocking. Flocking is a behaviour which is often described with birds. Birds that fly in a group have a shared destination they want to go to. This destination can be reached by flying in a group. The group is formed through flocking. The actual flocking is a combination of aligning, cohesion & separation. The goal of this demo is to represent flocking behaviour.   
  
The demo in its current state is more of a simulation than a game, but the player has some interaction within the game. This turns the simulation somewhat into a game, which can be played. The simulation itself can still be seen even if the player does not do anything.

The objective I have set for this demo is that I want a group of animals to follow one leading animal, e.g. lion tribe. The animals following the leading animal would behave as birds do when they are flocking. The effect created is flocking, and it is amazing and very pleasing to look at. The animals can be controlled by the player. The player has arrow keys on the keyboard which the player can use to determine the leaders locations.

Design

The technical requirements for this demo to work means that we need to figure out how flocking actually works, and how to implement this behaviour in code. On the internet there are numerous articles about flocking, boids, bird behaviour etc. The pictures below show how flocking works, and I will explain what these pictures mean below the pictures.

Alignment Separation Cohesion

Pictures (Reynolds, 1995).

The Alignment part makes sure that the unit that needs to flock is aiming towards the average looking direction of the group. The unit takes the average facing direction of the units, and puts that average into a vector, which can be combined with a certain speed to create a force towards that average direction the units are facing.   
  
The Separation part means that the units do not want to be close to each other, so they have to keep their distance from each other. I did not implement the separation in code, but instead put hitboxes on the units, so they would not go through each other, and have a certain distance from each other, even though there is no code for it.

Lastly cohesion makes sure that the units move towards the centre of the group. This means that the average location of the units around the unit are calculated, and put into a variable. This variable will then be taking into account when using other vectors, to make sure the unit knows where to go.

For the separation part of the flocking I chose not to implement it by code, but to just add a hitbox on the units, making sure that the units do not overlay each other.   
  
An extension of the demo would be to actually implement the separation in code, and to maybe add an Artificial Intelligence that can determine a specific route which the following animals can take. So the game would create a random destination goal, and the leader of the pack will determine a certain route, and the following units will follow the leader towards the destination.   
  
Development/Demo  
  
I am quite new to programming in Unity (I started this year), so I used a few tutorials to help get me started with the code. After looking at the tutorials, I was able to implement the flocking behaviour in code, which I now understand. The thing with me and programming is that I know the steps, the process, but I do not know how to implement it in code. I was very pleased when the code worked, because I had my doubts.

The technique is quite simple, but hard to implement for me.  
First, there should be leaders of the packs. I have implemented two packs of 40 units, and each pack has its own leader. The leaders are actually game managers, which manage the units around itself. The leaders are the goal of all the units combined, and the location of the leaders is taken into account when determining the vector of the following units.   
  
To make sure the alignment is correct, the unit processes the distance from another unit on the screen (random), and checks if the distance is higher or lower than a certain value. If the value is higher, the unit does not change its direction, because the second unit is not considered its neighbour. On the other hand, if the value is lower than the certain value given, the unit is counting the other unit as its neighbour. If this is the case, Add the velocities together, and normalize this value, to get a vector in the average direction the two units are travelling towards. In the picture on the previous page you can see how this can be represented. The two lines in front of the green triangle mean the direction it has (green), and the direction it will take (blue). The blue one is calculated with the above mentioned steps.

To make sure the cohesion is correct, the unit processes the distance from another unit on the screen (random), and checks if the distance is higher or lower than a certain value. If the value is higher, the unit does not change its direction, because the second unit is not considered its neighbour. On the other hand, if the value is lower than the certain value given, the unit is counting the other unit as its neighbour. If this is the case, add the locations together, and normalize this value, to get a position of the average units, and travel towards this new location stored in a different variable. In the picture on the previous page you can see how this can be represented with the green lines coming out of the other blue units. The average location of the four blue units and the green unit are combined together into a new value, normalized to get a vector length 1, and are given to the green unit to process the location it wants to travel towards.  
After the cohesion, separation and alignment values are calculated, the values will be added, and stored in a new variable. This variable is a vector, and this vector is normalized to give every unit the same amount of force. Then the force is multiplied by a certain speed, and added to the unit, and the unit travels in the direction calculated.

I could not really find other alternatives to implementation in different ways, other than just using hitboxes. I have used this myself with the separation, but not on the other two parts of flocking.

Evaluation  
  
The demo has two main animals, the leaders, which are static. The player can move the leaders at the same time with the arrow keys. The other animals will follow the leader, and eventually all clump together. They will not clump together in a perfect circle, but make sure that the demo is never the same. Sometimes, a grey unit will get stuck between a group of orange units, but moving the leaders to a different location may slingshot the group away, which causes the stuck unit to get free, and follow its own leader again successfully.  
The demo met the requirements I set, but if there was something I would change, it would be to be able to move the two leaders of the groups separately, or to have the leaders figure out a path on their own.   
  
Another cool addition to this demo would be a few buttons and settings on the screen itself, and a start button. The buttons and settings on the screen would mean amount of units, if the units have a goal or not, if they are willing to take their own path, and if they want to stick together, or avoid each other. This would turn the game into more of a simulation, but if you compare the game with the analysis and the design, it would add a lot more to the assignment and the goal of the demo.   
  
A nice addition to this would be that there is some kind of predator which chases the group and the leader, which results in the group and leader fleeing, seeking for a certain safehouse. The safehouse could be a position the player can set, so the player has an influence in the game, so the game is actually a game, and not a simulation.   
  
The separation part has not been hardcoded in the game, but is a result of hitboxes colliding with each other, and therefore units bumping each other aside. For further improvement of the game I would like to code this part, and make sure that the units do not clump together as much as they do now.

# Bibliography

Reynolds, C. (1995, June 29). *Boids background and Update*. Opgehaald van red3d: https://www.red3d.com/cwr/boids/