CSP2307 Introduction to 3D Game Programming

Workshop 7: The bobcat is my shepherd

This workshop is worth 4 marks, to be demonstrated in the next workshop

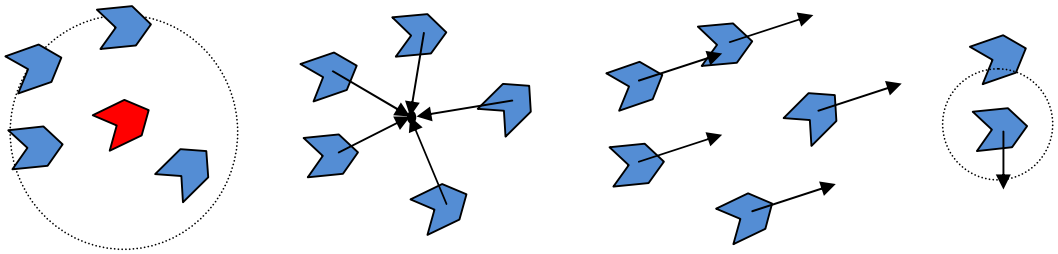
Outcomes:

By the end of this workshop you should be:

- Familiar with vector addition, determining a vector between two points, and conversion from polar to Cartesian coordinates.

Background:

In 1986 Craig Reynolds developed a way of modelling the behaviour of flocks of birds – known as 'boids'. The behaviour of each member of a flock can be modelled using several rules. These rules are simple, but when enacted by a large number of individuals lead to seemingly complex "emergent" behaviour.



Cohesion **Alignment** **Separation**

An individual belongs to a 'flock' as defined by some area around that individual (eg. Other individuals closer than some distance away). The individual is influenced by a number of forces:

(a) A force that pulls the individual towards the centre of the flock (cohesion).

(b) A force that tries to move the individual in the direction that the flock is heading (alignment).

(c) A force that prevents the individuals from 'crowding' each other (separation).

These same behaviours can be used to model schools of fish and animal herds. In this workshop you will implement this behaviour for a flock of sheep – with additional repulsive and attractive forces provided by objects in the world.

Preparation:

You have two options, either follow the instructions below to retro-fit any of the previous workshops (even as far back as the original bobcat code). Or download the full solution to the past workshops and build from there

If you are retro-fitting your code:

Take code from any previous working workshop and add the workshop files that you can download from blackboard.

- **GameController.h** should go in the root directory with the other C++ files
- The **sheep** folder should go into the **resources** folder.

Open the workshop in Visual Studio, and add GameController.h to the project.

In winMain.cpp, #include the GameController.h file and make the following modifications:

1) Use a 3rd person camera that follows the excavator x-position (eg. As in the previous workshop)
2) In the game loop – remove the code that computes the game mechanics and replace it with a call the GameControllers **runGameLogic** method. The new game loop should look like this:

```
while (exit==false) // while we dont want to exit…
   {
         exit = gameWindow.messagePump(); // process …

         bool status = bigExcavator.processKeyboardInput();

         GameController::runGameLogic(worldThings);

         // render the world
         renderer->Render(bigExcavator, worldThings);
   }
```

Compile and run the code – the police cars should now stand still.

3) Modify the **generateMap** method. Get rid of the police cars, leave just one house, and add a number of sheep. Eg. the code for adding the house and sheep:

```
// add a house to the level
Geometry house;
house.setName((string)"house");
house.loadGeometry("resources\\simpleHouse.ASE");
house.setColour(1.0f,1.0f,1.0f);
house.setPosition(Vector3f(150.0f,0.0f,0.0f),(string)"house");
worldThings.push_back(house);

// add sheep
Geometry sheep;
sheep.setName((string)"sheep");
sheep.loadGeometry("resources\\sheep\\sheep.ASE");
sheep.setColour(1.0f,1.0f,1.0f);
for (int count=0; count<20; count++)
{
sheep.setPosition(Vector3f(count,0,count),"sheep");
sheep.setID(count);
worldThings.push_back(sheep);
}
```

Compile and run – you should get a group of sheep, moving diagonally down – but not exhibiting flocking behaviour.

Task 1 (3 Marks):

To implement flocking, you will modify the **updateSheep** method in the GameController class in between the 'your code start here' and 'your code ends here' markers. You fill find the Vector3f class very useful, so examine it and its methods.

For the sheep – find the cohesion, alignment, and separation vectors – add them to the heading vector.

To do this, loop through the vector of game objects, and if the object is a different sheep determine the distance between the two sheep, and if it's less than FLOCKRADIUS compute the following:

- Add that sheep's position to a sum of sheep positions.
- The direction the sheep is heading is stored as an angle – convert this angle to a vector (use the cos and sin relations). Add this vector to a sum of headings.
- Increment a variable that holds the number of sheep in the flock
- If the distance between the two sheep is also less than some crowding distance, add a force to the heading vector that will separate them – make this force inversely proportional to the distance between the sheep. (**this is the separation force**)

After the loop is done, divide the sum of sheep positions by the number of sheep to find the centre of the flock, and add a vector from the sheep to this centre to the heading vector (**this is the cohesion force**).

Also, normalize the sum of headings and add this to the heading vector (**this is the alignment force**)

Task 2 (1 Mark):

Modify the code further, so that the excavator repulses the sheep when within a certain range defined by BOBCATRANGE, and that the sheep are attracted towards the house when within a certain range defined by HOUSERANGE.