Prof. Dr.-Ing. Ralf Steinmetz
Multimedia communications Lab

Dr. Florian Mehm
Dipl. Inf. Robert Konrad

Game Technology
Winter Semester 2016/2017

Exercise 13

For bonus points upload your solutions until **Saturday, February 4, 2017, 9:50**

General Information

- The exercises may be solved by teams of up to three people.
- The solutions have to be uploaded to the Git repositories assigned to the individual teams.
- **The submission date (for practical and theoretical tasks) is noted on top of each exercise sheet.**
- If you have questions about the exercises write a mail to game-technology@kom.tu-darmstadt.de or use the forum at https://www.fachschaft.informatik.tu-darmstadt.de/forum/viewforum.php?f=557

P13. Practical Tasks: AI & Sound (5 Points)

In this exercise, we program a simple world for artificial intelligences to work and make sound in. The first part of the AI task is building a part of the flocking algorithm to control boids in the world. The second part allows an AI to switch between different steering behaviors based on the current game state.

The code is provided for you, your task is to fill out the respective functions. The code can be found at https://github.com/TUDGameTechnology/Exercise13

Please remember to push into a branch called "exercise13".

P13.1 Flocking – Cohesion/Seek and Separation/Flee (1 point)

In `Flocking.h/.cpp`, you can see that cohesion and separation for flocking can be implemented by seek and flee steering behaviors. You can find the (empty) implementations for seek and flee in `Steering.h`. Implement these functions.

Seek should accelerate at maximum acceleration towards the target position (= the output to the "linear" component of the steering output should be a vector pointing towards the target with magnitude `maxAcceleration`). Flee is the opposite of this behavior.

Note: Without these functions, the moon's behavior will do nothing and the boids will slow down and stop eventually.

P13.2 Utility-based Reasoner (2 points)

The moon object has three options:

- Wander: In this option, it wanders randomly around
- Seek: In this option, it seeks towards the position of the earth
- Continue Seek: This option is added to force the moon to commit for a certain time to the seek. It uses the opt-in pattern to make sure this option gets chosen right after the Seek option has finished executing.

The default option is Wander. If it is closer than 1 unit to the Earth object, the moon should switch to the Seek option.

The two options have actions that are steering behaviors. For Wander, the moon AI has a wander steering behavior. In the option Seek, it has a seek steering behavior, with the Earth object as its target.

The three options and corresponding considerations and curves are already implemented (you can find them in `InitAI()` in `Exercise.cpp`. Note: It should be apparent from the source code why it is preferable to have editors for setting up utility-based AI ;-)

Your task is to implement the missing code for choosing which option to execute in the function `EvaluateOptions` in `Reasoner.cpp`.

P13.3 Sound (2 points)

In this part of the exercise, we implement basic positional audio. The listener is the Earth object, the sound source is the Moon object.

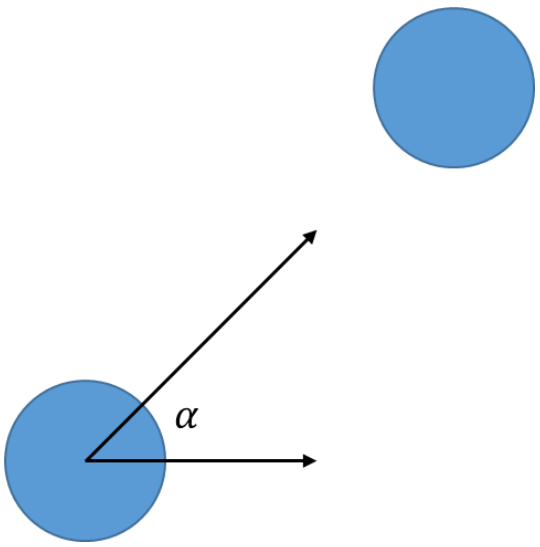
You can find the relevant source code in the class `DynamicSound` in `DynamicSound.cpp`. The sound is provided as a 16 bit interleaved stereo stream. In the original array, you find the unmodified version of the sound. In this array, all values with even array indices are for the left side, all values with odd array indices on

the right side. Your task is to modify the sound in realtime to account for the relative position of the listener (the Earth) and the sound source.
This includes two tasks:

1) Mixing between left and right ear

To implement this, you can use the cosine of the angle between the listener and the sound source. Use it to set the values of two float values which indicate the relative intensity of the sound. If the sound is very close to the listener in the horizontal direction, choose equal factors of 0.5 and 0.5.

Hint: You can normalize the direction vector and use the x-axis for this. See the following figure for an illustration of this. If the second sphere is straight ahead, the cosine will be 0, and if the second sphere is directly to the right of the sphere, it will reach its maximal value.



2) Distance attenuation

The sound should get fainter the further away it is positioned. You can do this by dividing the mixed sound value by the exponential function of the distance.
Combining 1) and 2), the value of the sound sample should be:

$$x' = x \cdot \frac{volume}{e^{distance}}$$

T13 Theoretical Tasks: AI & Sound (5 Points)

T13.1 Steering behaviours (2 points)

You are given two objects, A and B, with the given positions and speeds. Fill out the following table of steering outputs if A is given B as its target. The steering output could be requested as a velocity or an acceleration. In both cases, it is a two-dimensional vector.

The maximal speed of A should be 2, the maximal acceleration of A should be 0.5.

Object	Location	Velocity
A	5, 6	3, 3
B	8, 2	3, 4

Steering behavior	Steering output (vec2)
Seek (kinematic)	
Flee (kinematic)	
Seek (dynamic)	
Flee (dynamic)	

T13.2 Movement algorithm identification (1 point)

Consider the provided pseudo-code for a movement behavior:

```
vec2 GetSteering(character c) {
      // Get orientation in degrees
      float orientation = c.orientation
      float randomDelta = randomInRange(-20, 20)
      float newOrientation = orientation + randomDelta
      return vectorFromAngle(newOrientation) * maxSpeed
}
```

a) Is this behavior kinematic or dynamic? Explain your answer. (0.5 points)

b) Which of the behaviors presented in the lecture is realized in the pseudo-code? Explain your answer. (0.5 points)

T13.3 AI & Sound (0.5 Points)

You implement a game in which AI characters can speak by playing back recorded sound files. Your AI uses a utility-based reasoner. You want to ensure that the option for playing the sound remains active until the sound file has finished playing. How can you achieve this when using Utility-Based AI?

T13.4 Doppler Effect (1 Point)

Consider a car driving with 150 km/h and a person running away from the car with 15 km/h (the two move in the same direction). The car emits lots of different sound effects. How much does the frequency of those sound effects change for the person when the car passes him or her, i.e. how large is the maximal difference between the frequencies?

T13.5 Sound Synthesis (0.5 Points)

How is wavetable synthesis carried out?